VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

WEBOVÝ PORTÁL PRO PŘÍSTUP KE GENERÁTORU VHDL KÓDU

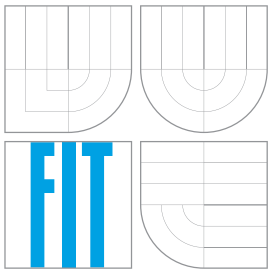
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR POUPĚ

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

WEBOVÝ PORTÁL PRO PŘÍSTUP KE GENERÁTORU VHDL KÓDU

WEB PORTAL FOR VHDL CORE GENERATOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR POUPĚ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MARTIN STRAKA

BRNO 2010

Abstrakt

Bakalářská práce se zabývá vývojem portálu pro obsluhu generátoru VHDL kódů. Představuje dosavadní dostupné možnosti generování hlídacích obvodů. Důkladněji se zaměřuje především na obsluhu generátoru pomocí webového prostředí a jeho následného využití ze strany uživatelů a popisuje celý vývojový cyklus od analýzy, specifikace a implementace až k testování vytvořené aplikace. Práce má za výsledek systém založený na skriptovacím jazyce PHP a databázi MySQL, který umožňuje uživatelům spravovat celé projekty.

Abstract

In this Bachelor thesis, activities which aim at developing web portal for an intuitive access to VHDL core generators are presented. The basic principles of methodology for generating VHDL descriptions of hardware checkers for communication protocols and RTL circuits are demonstrated together with their impact on the fault tolerant architectures. The main goal of this work is to develop user–friendly web environment which would facilitate the use of VHDL core generators. The specification and features of web portal are described together with implementation details and testing of final application. As the results of this thesis, the web portal based on PHP and MySQL database was created.

Klíčová slova

Webový portál, generátor VHDL, hlídací obvod, systém odolný proti poruchám, spolehlivost, VHDL, PHP, Python, HTML, CSS, JavaScript, MySQL.

Keywords

Web portal, VHDL generator, checker design, fault–tolerant system, dependibality, VHDL, PHP, HTML, CSS, JavaScript, MySQL.

Citace

Petr Poupě: Webový portál pro přístup ke generátoru VHDL kódu, bakalářská práce, Brno, FIT VUT v Brně, 2010

Webový portál pro přístup ke generátoru VHDL kódu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Straky

.....
Petr Poupě
17. května 2010

Poděkování

Mé poděkování patří dvěma osobám. Zaprvé vedoucímu práce, Ing. Martinovi Strakovi za jeho odbornou pomoc a čas, který věnoval konzultacím. Druhé poděkování patří slečně Lence Galbové za jazykovou korekturu.

Obsah

Kapitola 1

Úvod

„Vše, co se porouchat může, to se porouchá!“

To tvrdí jeden z Murphyho zákonů a každý z nás má jistě dostatek důkazů o tom, že to platí pro každý obor lidské činnosti. Zůstává jen otázkou kdy.

V roce 1991 bylo v Perském zálivu zabito 28 lidí a 97 zraněno díky drobné chybě. Řídicí systém obraných raket Patriot pracoval bez přestávky více než 100 hodin, na což nebyl testován. Při zaokrouhlování hodin během ukládání do paměti došlo k poměrně malé odchylce oproti reálnému času. Díky tomu však radar hledal střelu při útoku téměř o 700 metrů dál, než skutečně byla, a systém neměl šanci uspět při obraně vojenské základny.

V dnešní době je náš civilizovaný život závislý na strojích všeho druhu. Ať už jsou to stroje mechanické, či elektrické, spoléháme na ně natolik, že v případě jejich poruchy dochází ke ztrátě financí, poškození životního prostředí nebo při nejhorším ke ztrátám na životech. Mělo by tedy být samozřejmostí, že spolehlivost dostatečně zajistíme, ale děje se tak i v případě elektronických součástek, které jsou základním kamenem dnes již většiny zařízení? Návrh spolehlivého systému je věcí poměrně složitou, a tak stále ještě výsledná spolehlivost záleží hlavně na financích.

Spolehlivost spolu s odolností proti poruchám (Fault Tolerant – FT) tvoří významnou kapitolu při navrhování číslicových obvodů. V praxi se běžně používá pro zlepšení spolehlivosti systému replikace funkčních jednotek a následné vyhodnocování paralelně zpracovaných výstupů. Zde se však potýkáme se zvýšenými nároky na zdroj obvodu a zvýšenou spotřebou energie, což nemusí být vždy přijatelný faktor. Pokud vyvíjíme systém odolný proti poruchám, setkáváme se dále s vyššími nároky na návrh řízení a testování obvodu, proto je vhodné stále zkoumat další metody testování obvodu a metody pro návrh systémů odolných proti poruchám [8].

V rámci výzkumných aktivit na Vysokém učení technickém v Brně byla navržena kompletní metodologie pro tvorbu systémů se zvýšenou spolehlivostí využívající techniky odolnosti proti poruchám založené na bázi obvodů FPGA. Pro zvyšování odolnosti systému proti poruchám se doposud nejčastěji využívaly techniky TMR a duplex nebo bezpečnostní kódy [3]. Hlídací obvody mezi těmito technikami nenašly příliš velké použití. To se však mění s návrhem nové metodologie, která se opírá o využití hlídacích obvodů v různých architekturách odolných proti poruchám. Nejnižší vrstvu výzkumu tvorby FT systémů tvoří metodologie založená na automatizovaném vytváření hlídacích obvodů pro testování správnosti komunikačních protokolů [9].

Tato práce vychází z generátoru hlídacích obvodů pro komunikační protokoly a číslicové obvody popsané na úrovni meziregistrových přenosů RTL (Register Transfer Level). Vygenerované obvody následně slouží pro testování korektní funkce obvodů. Dále je práce

založena na generátoru TMR a duplexních architektur popsaných pomocí HDL (Hardware Description Language).

Hlavním cílem je zpřístupnění těchto generátoru návrhářům pod rozvinutou webovou aplikací. Webový portál by rozhodně neměl sloužit pouze jako prostředník mezi uživatelem a generátorem a jen skládat požadavky do příkazové řádky. Aplikace musí poskytnout uživateli plnohodnotné pracovní prostředí, kde může vložit celý svůj projekt a na konci očekávat rozšířený projekt o obvody odolné proti chybám. Výsledné pracovní prostředí by mělo nabízet uživateli možnost vkládat své úpravy v každé části tvorby projektu.

Výhodou komplexního webového portálu je efektivní a rychlý přístup k projektu s možností jednoduché správy ze strany administrátora portálu. Celý proces tvorby hlídacích obvodů se tak obejde bez instalace nového softwaru a nutnosti studovat novou problematiku, která je popsána v následujících kapitolách.

Druhá kapitola je věnována systémům odolným proti poruchám a principům použití hlídacích obvodů. Analýzou a specifikací požadavků na výsledný systém se zabývá kapitola 3. Celkový návrh systému a detaily funkčnosti aplikace popisuje kapitola 4. Implementační detaily v kapitole 5 jsou následovány kapitolou 6, která seznamuje s výsledky testování. Vzhledem k několika možnostem rozšíření aplikace se tomuto tématu věnuje kapitola 7. Bakalářskou práci uzavírá osmá kapitola, kde jsou shrnuty získané poznatky a uveden směr dalšího vývoje aplikace.

Kapitola 2

Architektury odolné proti poruchám

V úvodu byla rozepsána motivace tvorby aplikace a v této kapitole se seznámíme s teorií, která je důležitá pro pochopení funkčnosti obsluhovaných programů. Uvedeme metody, které zajišťují rezistenci proti chybám, jež se mohou v navržených architekturách vyskytovat. Taktéž se obeznámíme i s dostupnými nástroji pro generování hlídacích obvodů v architekturách odolných proti poruchám, tzv. Fault Tolerant systémech.

2.1 Metody pro zajištění odolnosti architektury

Termín spolehlivost se zabývá pravděpodobností, zda se systém v daném čase porouchá, což znamená, že výstupy systému nebudou správné. Avšak chyby, které výstupy negativně ovlivňují, jsou v elektrotechnice velmi častým jevem. Při navrhování odolné architektury je potřeba s nimi počítat.

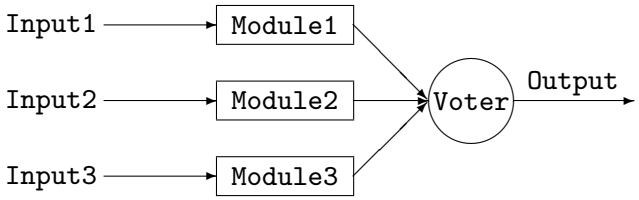
Při nahrazování mechanických jednotek elektronickými (rozdělovač, regulátor, apod.) došlo k výraznému snížení spolehlivosti systému. Pokud ale chceme spolehlivost zachovat, je třeba vytvářet systémy, které mohou dále provádět zadanou úlohu za přítomnosti obvodových poruch a programových chyb. K tomu existují následující tři základní metodiky, které pomáhají eliminovat vliv chyby na systém: předcházení chybám (fault avoidance), maskování chyb (fault masking) a eliminování vlivu chyby (fault tolerance). Tyto metodiky zahrnují několik technik pro konkrétní zpracování eliminace chyby, kterým je potřeba se věnovat právě v místech, kde je jakákoli chyba nepřípustná. Příkladem může být vesmírný a letecký průmysl, železniční signalizační systém, medicínské a život udržující systémy či nukleární a chemické procesy. Pokud se v těchto systémech chyba vyskytne, je potřeba, aby ji řídicí systém zaregistroval, případně ji dovedl odstranit.

Každá technika pro eliminaci chyb skýtá nevýhodu v nadbytečné replikaci komponent, a následně navýšené spotřeby prostoru, energie či materiálu. Záložní systém vždy funguje jako náhrada za systém, ve kterém se vyskytla chyba. Při určování výsledné spolehlivosti nově sestaveného systému záleží na každé technice, jak záložní systém vyhodnocuje ve vztahu se systémem původním, stejně tak, jako na počtu záložních systémů. Je tedy nutné používat techniky ohleduplně a testovat, zda třeba i úspornější technika neplní stejný účel.

K zajištění spolehlivosti systému se nejčastěji používají techniky TMR a duplex [6].

2.1.1 TMR

Systém TMR je zkratkou z anglického Triple Modular Redundancy. Tento systém patří do skupiny systémů M *z* N, ve kterých je nutné ke správné činnosti systému jejich M prvků z celkových N prvků. Pod termínem TMR se rozumí paralelní zapojení tří prvků tak, aby výpadek jednoho vedl k maskování poruchy v systému.



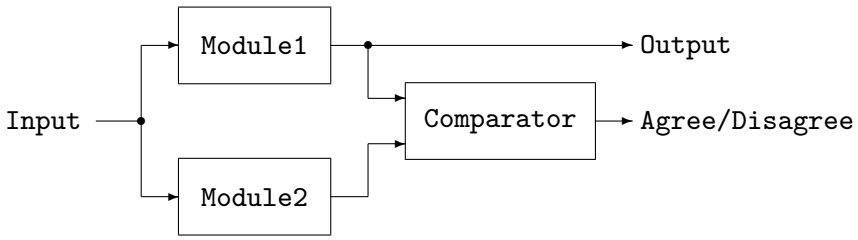
Obrázek 2.1: Schéma systému TMR.

Na obrázku 2.1, je zobrazen výstup od každé komponenty, veden na vstup rozhodovacího členu (Voter). Pokud se na jednom z členů vyskytne chybný výstup, rozhodovací člen jej sečte s ostatními nechybujícími výstupy a chyba se na výsledném výstupu neprojeví v plné míře. Systém může pracovat pouze s jedním chybným prvkem. V tomto systému se očekává od každého členu stejný výstup a při případném posouzení se zbylými prvky se navíc odhalí chybující prvek, který je možné následně rekonfigurovat.

TMR je běžně používáno ve vesmírné letecké elektrotechnice [1].

2.1.2 Duplex

Jedná se o techniku, využívající zdvojení komponent se srovnávacím členem na výstupu. Obrázek 2.2 popisuje strukturu zapojení. Oproti technikám M *z* N je zaveden chybový výstup, určující, zda porovnávané komponenty mají stejný, nebo rozdílný výstup. Pokud je zjištěn rozdílný chod, nebo případná porucha jedné z komponent, bude na chybový výstup odeslána informace o chybě. Z tohoto vyplývá, že systém je při výskytu chyby nefunkční.



Obrázek 2.2: Schéma techniky zdvojení se srovnávacím členem.

Zdvojení komponent zvyšuje bezpečnost zařízení a současně snižuje jeho spolehlivost. Tyto dva parametry mohou působit protichůdně, ale ve výsledku záleží na tom, pro jaké účely bude tato technika použita. V praxi použití této metody znamená zvýšení pravděpodobnosti nahlášení „planého poplachu“ za cenu vyšší bezpečnosti.

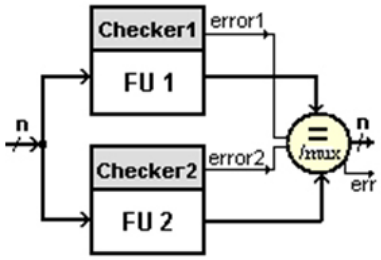
2.2 Hlídací obvody

Hlavní použití těchto prvků spočívá v kontrole komunikačních protokolů.

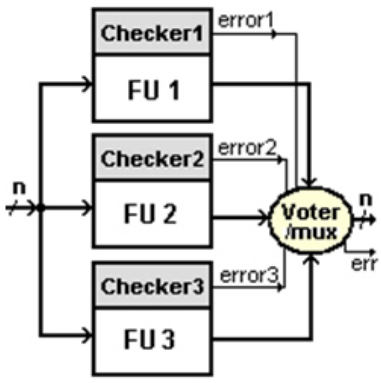
Hlídací obvod je založen na principu přídavného obvodu s obdobným rozhraním jako hlídaná komponenta. Vstupy i výstupy zůstávají zachovány, navíc je zaveden pouze chybový výstup. Výhodou tohoto výstupu je, že v případě výskytu chyby označí komponentu a je možné provést pouze částečnou rekonfiguraci v místě výskytu chyby. Hlídací obvody mohou být realizovány buďto jako vnitřní obvod, kde se pouze změní rozhraní přidáním chybového výstupu, nebo vnější obvod, který replikuje rozhraní původního a posuzuje chování podle předem stanoveného protokolu.

2.3 Aplikace hlídacích obvodů

Největší síla využití hlídacích obvodů je ve sloučení s jinou často používanou architekturou. V kombinaci s architekturami TMR nebo duplex vznikne mnohem odolnější systém proti chybám. Hlídač má schopnost detekovat a lokalizovat poruchu modulu. Klasický systém duplex v případě jednoho chybného prvku není schopen pracovat. Pokud je každý z prvků opatřen hlídacím obvodem jako na obrázku 2.3, architektura poskytuje správné výsledky i při poruše jednoho modulu. Taktéž se rozšíří spolehlivost systému TMR stejným použitím. Při sestavení architektury podle obrázku 2.4 vznikne systém, který je schopen vracet správné výsledky i při poruše dvou modulů.



Obrázek 2.3: Duplex architektura s využitím hlídacích obvodů.



Obrázek 2.4: TMR architektura s využitím hlídacích obvodů.

Hlídací obvod lze využít pro detekci poruchy v hlídané komponentě a k její lokalizaci. Tato funkce určuje hlavní směr použití hlídacích obvodů k tvorbě FT systémů. Mimo jiné

lze hlídací obvody použít pro verifikaci návrhu či k průběžné diagnostice.

2.4 Dostupné nástroje pro tvorbu FT systému

Pro účely webového portálu jsou využívány tři nástroje, které vznikly na UPSY — FIT VUT v Brně. K vygenerování výsledného obsahu jsou tyto programy jediným dostupným řešením.

Prvním programem je generátor hlídacích obvodů. Program je napsán v jazyce C a za vstup přijímá soubor s formální definicí komunikačního protokolu, nebo číslicového obvodu, pro kterou následně vytvoří hlídací obvod. Výstupem programu je již číslicový obvod v jazyce VHDL.

Dalším nástrojem vhodným pro účely portálu je program pro tvorbu TMR a duplexních architektur HDL. Tento program, taktéž napsaný v jazyce C, generuje několik nových obvodů z jednoho vstupního popisu obvodu. Zadáním parametru můžeme ovlivnit, zda se vygeneruje pět nových souborů pro systém TMR (tři kopie původní komponenty, rozhodovací člen a zastřešující komponenta), nebo čtyři nové soubory pro duplexní systém (dvě kopie původní komponenty, porovnávací člen a zastřešující komponenta).

Posledním nástrojem, který je nápomocen uživateli při tvorbě FT systému, je program — napsaný v jazyce C — pro převod VHDL kódu na formální popis architektury. Jaká je syntaxe výsledného formálního popisu, bude popsáno v kapitole 3.3. Za vstup přijímá VHDL kód zadané architektury a výstup je nově vytvořený soubor s příponou *.frm*, obsahující formální popis zadané architektury.

Všechny zmíněné programy jsou založené na práci s VHDL kódy, popisujícími pouze komunikační protokoly nebo číslicové obvody popsané na úrovni meziregistrových přenosů RTL. Pro jiné VHDL kódy nebude generování fungovat správně.

Kapitola 3

Analýza a specifikace požadovaného systému

Tato kapitola má za úkol zaměřit se na zkušenosti a potřeby potenciálního uživatele, který bude výslednou aplikaci využívat a analyzovat očekávání uživatele od aplikace tohoto typu. Mimo jiné si stručně představíme jazyk VHDL, ve kterém jsou napsány vstupní i výstupní kódy projektu, obsluhované některými programy.

Druhá část kapitoly se zaměřuje na specifikaci formálních popisů, se kterými bude uživatel přicházet do styku. Tato část je taktéž určena k vysvětlení syntaxe formálních popisů, pro případ potřeby uživatele přepsat či upravit část kódu z projektu.

3.1 Zaměření aplikace

Výhody webové aplikace jsou v dnešní době již téměř každému dobře známé. Vzhledem k vzrůstající rychlosti internetu na většině míst a zvyšujícímu se pokrytí pro připojení k internetu na cestách se mnoho aplikací přesunulo na webové portály. Například společnost Google zaměřila svůj vývoj na celý operační systém, který bude dostupný pro uživatele z internetu i s nastavením. Dříve bylo nemyslitelné přenášet přes internet větší objemy dat a vzdáleně s nimi pracovat. Dnes již ani toto není problém a existují i aplikace pro úpravy videa přes webové rozhraní.

Tato řešení nabízejí uživatelům mnohem větší flexibilitu z hlediska pracovního prostředí. Nemusejí se zabývat složitým nastavováním aplikace a v případě aktualizace instalovat novou verzi. Celý tento proces probíhá na straně serveru a uživateli je zprostředkováno pouze grafické rozhraní, za kterým je poskytnut i diskový prostor pro jeho práci.

Tímto směrem míří i tato aplikace. Všechny procesy proto musejí probíhat na straně serveru. Pro uživatele je zobrazeno pouze rozhraní, ve kterém zadá své požadavky, a po dokončení si může uložit výsledek své práce.

Přístup k aplikaci je očekáván nejčastěji od návrhářů číslicových obvodů, kteří budou potřebovat přístup ke generátoru hlídacích obvodů. Od těchto uživatelů se očekává, že jsou to zkušení uživatelé osobních počítačů a s prací na webových portálech mají bohaté zkušenosti. Což znamená, že není třeba ke každému kroku přikládat detailní popis toho, co má uživatel dále dělat. Je potřeba se zaměřit na jednoduchost a intuitivnost aplikace. Pokud bude uživatel přistupovat k aplikaci častěji, bude vyžadovat, aby práci dokončil během několika nezbytných kroků.

Při tvorbě aplikace je taktéž třeba dbát na přenositelnost aplikace na různé platformy.

Nasazení aplikace se může očekávat jak na veřejném portálu, tak i na portálu, který je dostupný jen z firemní, či školní sítě.

Uživateli bude nabídnuta možnost tvorby projektu, ve kterém bude docházet ke všem potřebným úpravám, v několika nejnutnějších krocích. Každý projekt předpokládá na vstupu archiv ve formátu *zip*, který obsahuje zdrojové kódy komponent ve formátu VHDL. Po určení volitelných prvků je uživateli nabídnut výstup znovu ve formátu *zip*, tentokráte již doplněný o kontrolní obvody ke zvoleným komponentám.

3.2 VHDL

VHDL (Very High Speed Integrated Circuits Hardware Description Language) je společně s Verilog HDL programovací jazyk pro popis číslicových obvodů a systémů. Důvodem ke vzniku tohoto jazyka byl problémový a prakticky nepoužitelný popis obvodů se stovkami tisíc hradel. Vyvinut byl jazyk kombinující rysy jazyka pro modelování a simulaci, jazyka návrhového, jazyka pro testování a jazyk pro popis topologie. Tento jazyk je užitečný při hledání chyb, kterých se mohl návrhář dopustit. Tomuto kroku se říká verifikace návrhu.[2]

3.3 Formální popis architektury

Formální popis architektury vychází z převodníku VHDL popisu na formální popis architektury. Výsledný popis je v bakalářské práci podkladem pro vygenerování všech prvků, u kterých uživatel bude volit možnosti generování hlídacích obvodů. Pokud bude VHDL kód popisovat číslicový obvod, složený z dalších obvodů, musí být vnitřní struktura vnořeného obvodu zapsána v samostatném souboru s VHDL kódem.

Pro popis takto popsaných protokolů a číslicových obvodů byl vytvořen formální popis architektury. Ten v sobě zahrnuje informace o zastřešující entitě i seznam všech vnitřních komponent. Také je v popisu zapsán seznam propojení — jak propojení vnější se zastřešující komponentou, tak i propojení vnitřní, mezi komponentami, tvořící obsah komponenty hlavní.

Následující kód je vysvětlením syntaxe formálního popisu, který se ukládá do souboru s příponou `.frm`

```
[ENTITY name(level) {all IN/OUT/INOUT ports}
components_list,
signals_list,
connection_list,
info]
```

Tato syntaxe definuje entitu a primární rozhraní obvodu. Seznamy komponent `component_list`, signálů `signals_list` a propojení `connection_list` mají vlastní syntaxi, popsanou v následujících příkladech:

```
** Definice všech prvků entity - components_list
COMPONENT name(level) {all IN/OUT/INOUT ports}

** Definice všech spojovacích signálů entity - signals_list
SIGNAL(name,type,bit_width,constant_value)
```

```
** Definice spojovacích signálů mezi vnitřními prvky - connection_list
Type_of_connect(bitwidth,signal,component1,component2,outport1,inport2)
```

kde hodnota `Type_of_connect` je jedním z vyjmenovaných klíčových slov: `PRIMINCON` (pro spojení vnitřních prvků se vstupy zastřešující entity), `INTERCONNECT` (pro spojení vnitřních prvků) nebo `PRIMOUTCON` (pro spojení vnitřních prvků s výstupy zastřešující entity.

V definici komponenty nalezneme seznam definicí portů {`all IN/OUT/INOUT ports`}. Každá položka tohoto seznamu má následující syntaxi:

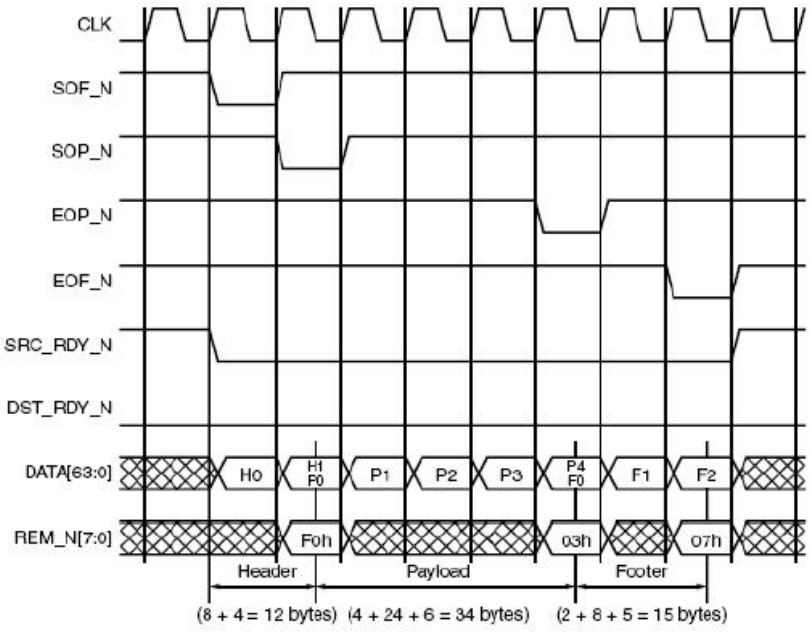
```
IN/OUT/INOUT(name,type,bit_width,constant_value)
```

Každý příkaz je oddělen znakem konce řádku. Komentáře jsou uvozované znakem „`*`“ a ukončovány koncem řádku.

3.4 Formální definice chování komunikačního protokolu a číslicového systému

Generátor hlídacích obvodů potřebuje na svém vstupu popis komunikačního protokolu. Jeho syntaxe je detailně popsána v [10]. V této kapitole si na příkladu vysvětlíme možnosti použití tohoto popisu.

Jako příklad byl zvolen protokol LocalLink, vyvinutý pro propojování funkčních komponent v FPGA (Field-Programmable Gate Array). Časový diagram protokolu je znázorněn obrázkem 3.1, kde jsou označeny řídící i výstupní signály a jejich hodnoty v průběhu správné činnosti protokolu.



Obrázek 3.1: Časový diagram protokolu LocalLink.

Pro tento protokol byla vytvořena následující formální definice:

```
@
p0=SRC_RDY_N==0 and DST_RDY_N==0 and SOF_N==0 and SOP_N==1 and EOP_N==1
    and EOF_N==1;
p1=SRC_RDY_N==0 and DST_RDY_N==0 and SOF_N==1 and SOP_N==0 and EOP_N==1
    and EOF_N==1;
p2=SRC_RDY_N==0 and DST_RDY_N==0 and SOF_N==1 and SOP_N==1 and EOP_N==0
    and EOF_N==1;
p3=SRC_RDY_N==0 and DST_RDY_N==0 and SOF_N==1 and SOP_N==1 and EOP_N==1
    and EOF_N==0;
p4=SRC_RDY_N==0 and DST_RDY_N==0 and SOF_N==1 and SOP_N==1 and EOP_N==1
    and EOF_N==1;
p5=SRC_RDY_N==0 or DST_RDY_N==0;
#
(S0,p5):S0; (S0,p0):S1; (S1,p5):S1; (S1,p1):S2; (S1,p4):S1; (S2,p5):S2;
(S2,p2):S3; (S2,p4):S2; (S3,p5):S3; (S3,p3):S0; (S3,p4):S3;
$
```

V první části tohoto kódu jsou definovány proměnné (v kódu označeny `p`n), udávající kombinace hodnot signálů, které mohou ovlivnit přechod mezi jednotlivými stavy. Další část musí obsahovat dvojice stavů (stav je v kódu označen `S`n), rozšířené o proměnnou z předchozí části. V takto složené trojici je jako první uveden počáteční stav a následující stav se zapisuje jako poslední hodnota;

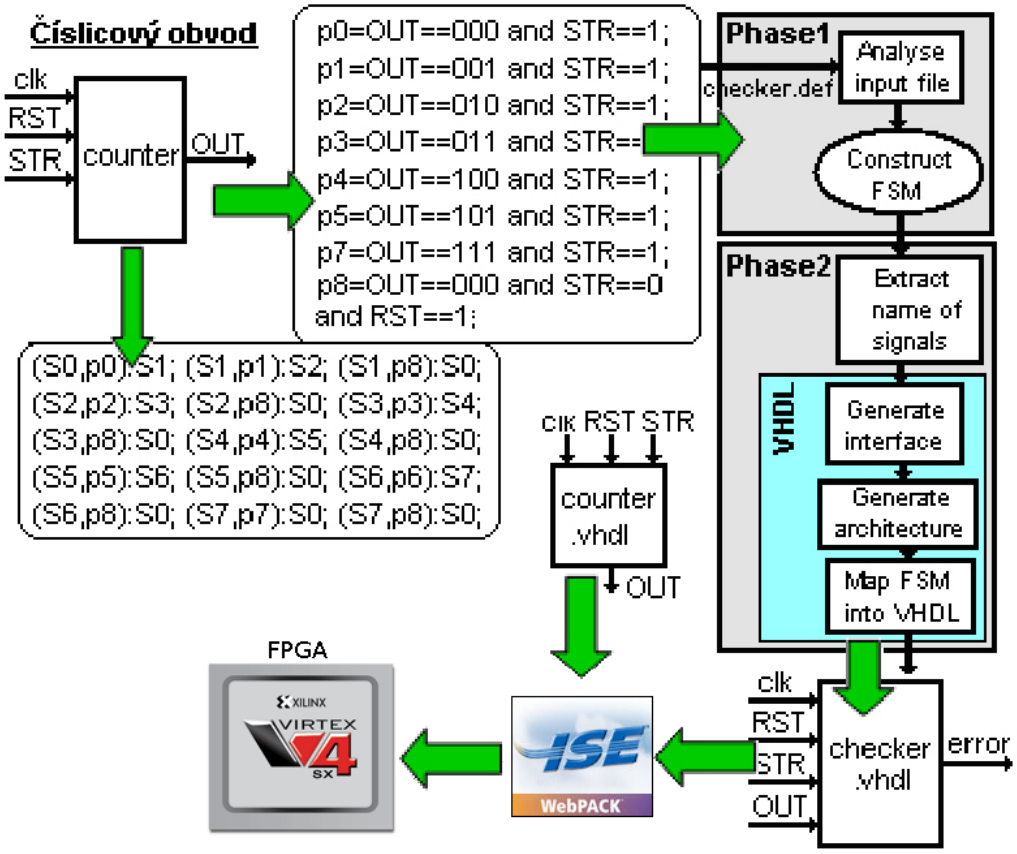
Formální popisy je možné vytvářet nejen pro komunikační protokoly, ale i pro číslicové obvody (např. čítač, dekodér apod.). Pro jednoduché číslicové obvody se nabízí možnost generování popisu automaticky v případě, že budou označeny typy každého ze vstupních i výstupních signálů. V aplikaci není možnost automatického generování formálního popisu a uživatel, pokud si nevybere z předdefinovaných popisů, jej musí sám vepsat nebo vložit ze souboru s příponou `.def`.

3.5 Princip generátoru hlídacích obvodů

Na sběrnici komunikačního protokolu v číslicových obvodech se mohou vyskytnout chyby, které je možné popsat různými způsoby. Kontrolují se jak chyby v kombinaci přenášených signálů, tak i chyby v jejich posloupnosti. Protože musí hlídací komponenta vykazovat sekvenční chování, je možné ji popsat pomocí konečného automatu. Na základě jazyka, který byl uveden v předchozí kapitole, je možné automaticky vytvořit hlídací obvod.

Princip funkčnosti generátoru znázorňuje obrázek 3.2. Stavy protokolu projdou analytickou kontrolou programu, který je následně v druhém kroku schopen vytvořit výsledný popis hlídacího obvodu, zapsaný již v jazyce VHDL. Vnitřní architekturou plně odpovídá konečnému automatu z popisu v definičním jazyce. Takto vygenerovaný hlídací obvod disponuje rozhraním původní komponenty a může se tedy začlenit do systému.

Hlavní výhodou tohoto přístupu oproti přímé implementaci je možnost generování obvodu bez přítomnosti zkušeného návrháře jen za pomoci jednoduchého popisu [9].



Obrázek 3.2: Princip generování hlídacího obvodu pro jednoduchý číslicový obvod.

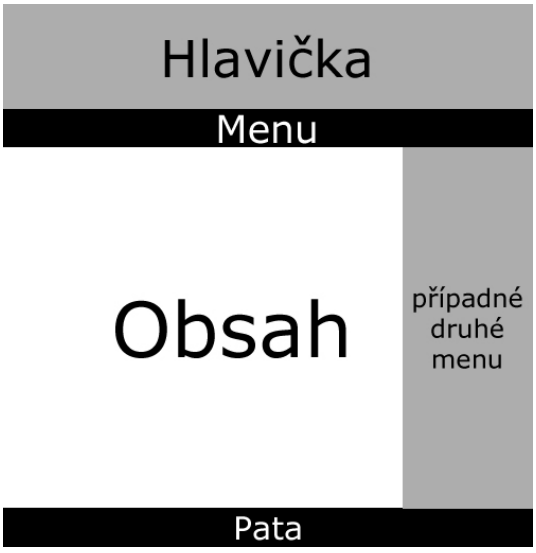
Kapitola 4

Návrh systému

Kapitola se zabývá vším, co bylo navrženo jako součást systému. Budou zde rozebrány podrobnosti ohledně celkové struktury aplikace a rozdělení návštěvníků podle práv. Jelikož je aplikace postavena na generování kódu, bude tato funkce znázorněna blokovým schématem, které vysvětluje činnost uživatele při tvorbě nového projektu. Následně jsou probrány i detaily pozadí aplikace, které tvoří samostatně běžící program pro obsluhu programů, generujících výsledné kódy. Pro objasnění ukládání dat nezbytných pro běh celého portálu bude odhalena databázová struktura, zvláště pak Entity-relationship diagram. Na závěr kapitoly bude zmíněn i diagram případů užití.

4.1 Struktura portálu

Účelem portálu je zprostředkovat přístup ke tvorbě projektu, čemuž musí být přizpůsoben i celkový vzhled. Není potřeba pro zkušené uživatele vyhrazovat místa v návrhu pro dodatečné nápisy s informacemi, které si uživatel může najít v technické zprávě. Z tohoto důvodu byl navržen vzhled aplikace podléhající struktuře naznačené obrázkem 4.1.



Obrázek 4.1: Schéma navrženého vzhledu portálu.

Hlavička, jakožto nejvýraznější prvek na stránce, v sobě zahrnuje i formulář pro přihlášení

uživatele — po přihlášení je na tomto místě zobrazena informace o nových zprávách spolu s tlačítkem pro odhlášení. K hlavičce je připojeno horizontálně rozložené menu, kde jsou hlavní kategorie portálu uspořádány podle důležitosti a četnosti používání. Navíc je zde nabídnuta i možnost vytvoření nového projektu, která svojí důležitostí může tvořit samostatnou kategorii. Do obsahu, který je umístěn pod menu, je zavedena možnost přídavného vertikálního menu, které má za účel zpřístupnit jakýkoli hlavní obsah nejvýše do dvou kliknutí a usnadnit uživateli orientaci v obsahu.

Při návrhu vzhledu aplikace byl brán zřetel na výslednou šířku zobrazované pracovní plochy aplikace. Limit šířky vzhledu byl určen na tisíc obrazových bodů, což je počet vyhovující dnes již většině monitorů.

Při první návštěvě portálu se uživateli zobrazí pouze základní obrazovka s uvítacími informacemi a možností přejít na stránku, kde se o projektu dozví veškeré potřebné informace. Pro jakoukoli interaktivní práci je potřeba se přihlásit pod uživatelským jménem a heslem. V portálu byla zavedena možnost automatického odhlášení po správcem nastavené době. Tato funkce běžně způsobí, že uživatel po přihlášení musí znovu vyhledat krok, ve kterém svoji práci ukončil. Tento problém je v portálu vyřešen odstraněním vlivu uživatelského stavu na adresu aktuální stránky a uživateli je po přihlášení zobrazena stránka, ze které byl pro dlouhou neaktivitu odhlášen.

4.2 Práva uživatelů

Přihlašování uživatelů je řešeno přes formulář vestavěný do vzhledu aplikace. Pro přístup na webový portál jsou rozlišovány tři základní skupiny oprávnění.

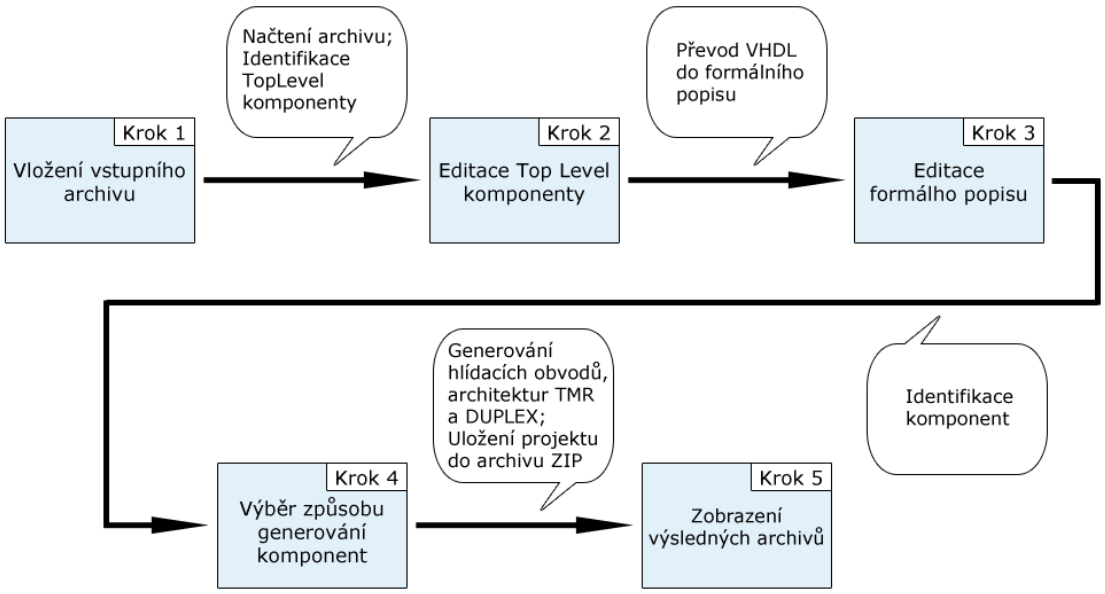
Do první skupiny bez práv spadají buďto nepřihlášení uživatelé, nebo uživatelé, kterým byla všechna práva odstraněna. Tito uživatelé mají přístup pouze k informacím o projektu a možnosti se buďto registrovat, nebo si vyžádat zapomenuté registrační údaje nutné k přihlášení. Odstranit všechna uživatelská práva je povoleno administrátorskému účtu, aby byla zachována možnost zakázání přístupu k účtu bez nutnosti jej mazat například v případě podezření na porušování pravidel provozu portálu.

Druhou skupinou oprávnění jsou přihlášení uživatelé, ti již mají plný přístup do aplikace, kde mají možnost vytvářet a editovat své projekty či číst systémové zprávy. Taktéž mají přístup k nastavení svého účtu, kde mohou změnit své registrační údaje, mimo e-mailu, nad kterým se při registraci provádí ověření uživatele. Registraci do této skupiny může provést kdokoli přes registrační formulář, ke kterému má přístup pouze nepřihlášený uživatel. Po vyplnění formuláře je nutné potvrdit registraci přechodem na odkaz odeslaný po dokončení registrace na zadaný e-mail. V sekci nastavení taktéž lze zvolit, zda zprávy o vygenerovaných kódech budou zasílány pouze v rámci systému, nebo budou odesílány i na e-mail.

Třetím oprávněním, které je potřebné v každém webovém portálu je administrátor. Toto oprávnění má v systému přístup pouze k editaci a vytváření uživatelských účtů. Pod tímto oprávněním lze uživatelům deaktivovat účet, změnit údaje či jim odesílat systémové zprávy buďto jednotlivě, nebo všem uživatelům s předchozími právy najednou. Administrátor však nemá možnost projekty vytvářet nebo přímo upravovat cizí projekty, což řeší způsob ukládání hesel všech uživatelů. Ta jsou uložena v otevřeném formátu a administrátor je může v případě potřeby k přihlášení se místo uživatele a provést potřebné úpravy. Hesla jsou uložena v otevřeném formátu z důvodu možnosti zaslání zapomenutého hesla na e-mail uvedený při registraci.

4.3 Blokové schéma aplikace

Těžištěm práce je tvorba projektu pro vytvoření hlídacích obvodů, která probíhá v pěti krocích. Po zvolení možnosti „vytvořit nový projekt“ může uživatel procházet těmito pěti kroky nezávisle na údajích, které v krocích vyplňuje. Pro základní orientaci v krocích aplikace slouží obrázek 4.2



Obrázek 4.2: Blokové schéma aplikace.

V prvním kroku se nejdříve zadávají základní informace o projektu, kterými jsou název a popis projektu. Druhou částí tohoto kroku je vložení archivu se zdrojovými kódy projektu, ze kterého bude projekt vycházet. Ve vloženém archivu může být mnoho komponent s různými jmény a není možné identifikovat hlavní komponentu, je tedy zapotřebí zadat jméno projektu stejné jako název souboru s hlavní komponentou, aby bylo možné tuto komponentu vyjmout zvlášť. Název je možné zadávat bez ohledu na velikost písmen a slova oddělovat mezerou i v případě, že se v názvu souboru jako oddělovač používá pomlčka či podtržítko. Jednoznačně identifikovat projekt je možné pomocí popisu, který se ve výpisu projektů zobrazuje přímo pod názvem projektu. Název nemá vliv ani na výsledné řazení projektů ve výpisu, protože řazení se řídí datem vzniku projektu, a tak se nabízejí uživateli v první řadě nejnovější projekty, u nichž se předpokládá pokračování v činnosti. Pouze v případě, že je soubor hlavní komponenty pojmenován `top_level.vhd`, je soubor rozpoznán automaticky a není třeba uvádět toto jméno v názvu projektu. Pokud vložený archiv nebude obsahovat žádný soubor s názvem projektu či bude–li archiv prázdný, přechod do dalšího kroku sice umožněn bude, ale omezí se tím funkce generování TMR a duplexních architektur. Taktéž v případě chybějících souborů pro komponenty, ze kterých se skládá hlavní komponenta, budou omezeny možnosti generování výsledných obvodů pro tyto komponenty.

Při přechodu na další komponentu dojde k otevření zadaného archivu a pokud obsahuje soubory s příponou `.vhd`, dojde k uložení těchto souborů do složky, která je vytvořena právě pro tento projekt. Taktéž pokud je rozpoznán název souboru hlavní komponenty, dojde k jeho uložení do souboru, který je možné v dalším kroku upravovat. Jestliže dojde poprvé k přechodu na další stránku či uložení, vytvoří se první verze projektu, do které se

ukládají změny ze všech dalších kroků.

Další krok nabízí možnost kontroly, zda byl z archivu načten správný soubor hlavní komponenty. Úpravy je možné uložit ve stávající verzi, nebo zvolit možnost „Uložit změny jako novou verzi“ a zkopírovat tak celý dosavadní projekt do nové verze, ve které se bude v dalším kroku pokračovat. Nahráním kódu ze souboru dojde k nahrazení stávajícího zobrazeného kódu, pokud nebude zvolena zmíněná možnost uložení změn v nové verzi. Pokud bude chtít uživatel pokračovat v předcházející verzi, musí přejít na zobrazení všech projektů a zde si u projektu vybrat verzi, kterou chce spravovat.

Po přechodu na třetí krok dojde k uložení změn ve VHDL kódu a vygenerování formálního popisu architektury. Zde byla zavedena možnost omezení velikosti vstupního souboru pro okamžité zpracování. Pokud je velikost souboru vyšší než právě povolená, dojde k zařazení požadavku do fronty ke zpracování. Díky této funkci se zamezí odmítnutí zpracování požadavků od jiného uživatele, který přistupuje k portálu ve stejný okamžik. Následně je po vygenerování kódu odeslána uživateli zpráva o úspěšném dokončení požadavku spolu s adresou pro přímé pokračování v projektu na místě úprav formálního popisu.

Pokud dojde k vygenerování formálního popisu, je tento kód zobrazen na stránce třetího kroku. V případě, že je předchozí krok dodatečně upraven, je potřeba zvolit možnost „Generovat znovu“ z důvodu uchovávání provedených změn formálního popisu. Formální popis je možné jak redukovat, tak i rozšířit o prvky, které v kódu chybí. Možnosti uložení jsou v tomto kroku stejné jako u kroku předchozího.

Po přechodu na čtvrtý krok aplikace dojde k vyjmutí rozpoznaných komponent z formálního popisu. Ke každé komponentě se definují patřičné signály a v archivu, který byl vložen v prvním kroku, se vyhledá, zda pro danou komponentu existuje soubor s VHDL popisem. Pokud uživatel přejde k tomuto kroku před vygenerováním formálního popisu, nebude mu ještě poskytnut výpis komponent, nýbrž jen hlášení o nedostupnost formálního popisu, ze kterého se přímo vychází.

Výpis rozpoznaných prvků je krokem, kde bylo potřeba zavést třídění komponent, aby měl uživatel stále přehled, kde na stránce najde požadovanou komponentu. Vzhledem k možnému velkému množství prvků je nutný i abecedně řazený seznam. Ten je umístěn nad výpisem komponent a je členěn podle zanoření (tzn. *level*) v zastřešující (Top Level) komponentě. Podle zanoření je tříděn i výsledný výpis komponent, u kterých se po rozbalení kliknutím zobrazí možnosti pro výsledné generování kontrolního obvodu.

V této nabídce má uživatel možnost vybrat typ výsledné kontrolní architektury, nebo kontrolní architekturu negenerovat. Tyto možnosti jsou předem určovány automaticky na základě rozpoznání typu komponenty z jejího názvu, uvedeného ve formálním popisu architektury. Pokud je rozpoznán typ komponenty, pro který je možné generovat formální definici potřebnou pro generování hlídacího obvodu, je automaticky označena možnost generování hlídacího obvodu a do pole pro formální definici se vloží patřičný předdefinovaný kód. Předepsané definice jsou zobrazeny dynamicky až po zvolení možnosti generování hlídacího obvodu z důvodu úspory místa ve výpisu komponent. Pokud uživatel nemůže zadat formální definici, může vybrat volbu generování TMR nebo duplexní architektury, a tak vytvořit FT architekturu i bez nutnosti použití hlídacích obvodů.

Zobrazení propojovacích signálů je obtížné implementovat mezi seznam komponent tak, aby byl výpis stále přehledný a zároveň bylo možné zjistit, se kterou komponentou je spojen zobrazený prvek. Tento problém byl vyřešen dvojicí tlačítek, umístěných ve spodní části každé zobrazené komponenty, které po najetí kurzorem zobrazí tabulku odkazů na připojené komponenty.

Po provedení změn je možné vše uložit přímo tlačítkem, které je u každé komponenty,

nebo nabídku znovu kliknutím na název komponenty zabalit a pokračovat v úpravách další komponenty. V případě, že v archivu s VHDL kódy chybí soubor pro danou komponentu, nebo nebyl její název správně rozpoznán z popisu Top Level komponenty, je blok s pořadovým číslem komponenty označen červenou barvou s vysvětlením, které se zobrazí po najetí ukazatelem na tento blok. Pro takovouto komponentu není možné generovat TMR ani duplexní architekturu. Při změně v předchozím kroku je potřeba zvolit možnost „Generovat znovu“, aby se zabránilo nechtěnému přepsání upravených voleb při přepsání či vygenerování nového formálního popisu.

Po přechodu ze čtvrtého kroku se uloží veškeré změny a dojde k zařazení požadavku o vygenerování všech potřebných kontrolních obvodů. Zpráva o dokončené práci generátoru je odeslána obdobně jako zpráva ve třetím kroku aplikace.

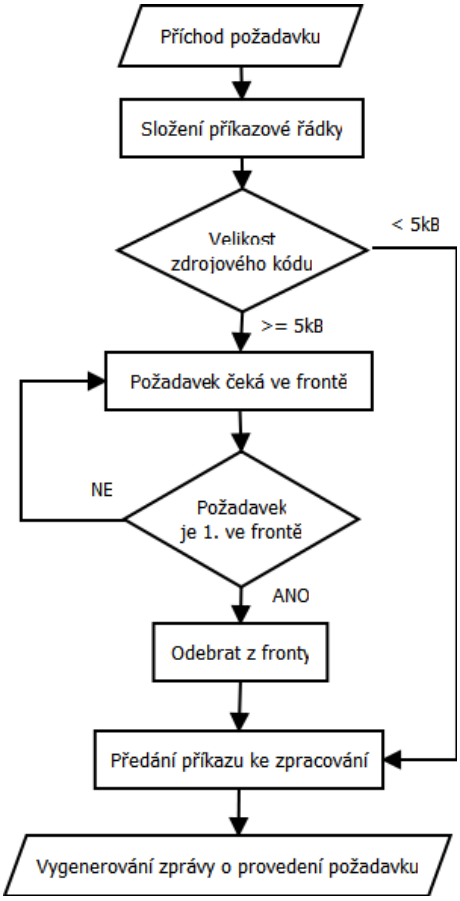
Pátý krok již pouze nabízí seznam s výsledným archivem celého projektu, stávající verze projektu a samostatnými soubory, u kterých byl zadán požadavek na vygenerování jedné z nabízených kontrolních architektur. V archivu s celým projektem jsou uvedeny složky jen s verzemi, které jsou dokončené (tzn. úspěšně provedeny až do pátého kroku a podařilo se vygenerovat archiv pro tuto verzi).

4.4 Generování kódů na pozadí

Při obsluze programu pro převod VHDL kódu na formální popis a generátoru hlídacích obvodů může dojít k přetížení serveru, pokud bude odesláno několik požadavků najednou od více uživatelů. Tomuto stavu je zabráněno vytvořením programu, který se stará o zpracování požadavků nezávisle na aplikaci, která je obsluhována uživatelem. Veškeré náročnější požadavky by prodloužily dobu načítání stránky, což by zabránilo uživateli v další práci na projektu. Ke zpracování požadavků na generování kódů byl navržen program zajišťující odebírání požadavků z fronty, přeposílání příkazů generátorům a následnou tvorbu zpráv o průběhu generování.

Vývojový diagram na obrázku 4.3 vysvětluje spolupráci portálu s programem na obsluhu požadavků pro generování kódů. Po příchodu požadavku se složí příkaz pro generátor s názvem vstupního, případně i výstupního souboru. Pokud je vstupní soubor větší než povolený počet bytů, nebo se jedná o hromadné provedení příkazů je požadavek uložen do souboru ve složce, která tvoří frontu k obsloužení. V případě jediného požadavku s menším vstupním souborem se příkaz okamžitě vykoná. Název souboru s požadavkem tvoří časová značka (anglicky Timestamp) s přidaným ID (identifikačním číslem) uživatele, který požadavek vytvořil, což zaručuje jedinečnost názvu souboru zároveň s možností odebírat soubory podle data vzniku. Tuto složku pravidelně kontroluje program napsaný ve skriptovacím jazyce Python.

Tento program musí být spuštěn na serveru neustále po instalaci portálu. Provádí stále jednu pracovní rutinu a uspí se na zadaný počet vteřin, pokud je kontrolovaná fronta prázdná. Činnost programu spočívá v odebírání požadavků z fronty a spouštění příkazů z požadavku. Pokud jsou všechny příkazy z požadavku zpracovány, uloží oznámení do složky pro dokončená generování a pokračuje zavoláním skriptu na obsluhu této složky, který je již součástí portálu.



Obrázek 4.3: Vývojový diagram obsluhy programů na pozadí.

4.5 Zasílání zpráv

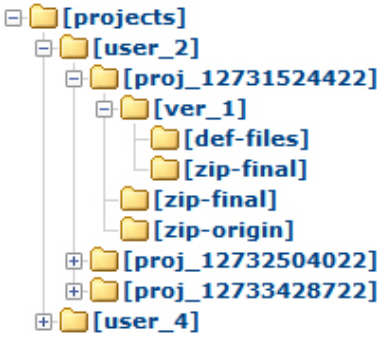
Oznámení o dokončení práce generátoru není nezbytně nutné pro funkčnost aplikace, která se při každém znovunačtení stránky naplní daty, která jsou již připravena pro další práci nezávisle na oznamovací zprávě. Pokud tedy ve frontě není žádný jiný požadavek, dojde ke zpracování v krátké době a uživatel po aktualizaci stránky již výsledek generování vidí ve svém projektu. S implementací systému pro zasílání zpráv však není nutné aktualizovat stránku několikrát za sebou v případě časově náročnější obsluhy požadavku. Uživatel má tedy možnost pracovat kdekoli jinde na portálu a počet nových zpráv je mu stále zobrazován v hlavičce každé stránky. U každé takto vytvořené zprávy je možnost přejít okamžitě na krok projektu, kde byl zadán požadavek pro generování.

Zprávy o vygenerování kódu mohou být odeslány zároveň na e-mail, pokud si uživatel tuto možnost zvolí v nastavení svého účtu.

4.6 Ukládání dat

Pokud pracujeme s aplikací, která interaktivně komunikuje s uživatelem a zároveň od něho přijímá obsáhlejší datové soubory, je nutné zajistit ukládání dat nejenom pomocí databáze, protože některé servery mohou mít problém s větším tokem dat z databáze. Aplikace pracuje se soubory od uživatele a také generuje nové vlastní soubory. Proto se ukládají všechny

soubory do adresářové struktury, která je naznačena obrázkem 4.4.



Obrázek 4.4: Adresářová struktura ukládaných souborů.

Pro každého uživatele, který si vytvoří projekt, se zavede složka, která je označena jeho identifikačním číslem. Tato složka obsahuje další složky se všemi dosavadními projekty daného uživatele, označené identifikačním číslem projektu. Ve složce projektu je složka jak pro obsah vstupního archivu s VHDL kódy, tak složka pro výsledný archiv celého projektu. Mimo těchto dvou složek pod projekt patří také složky všech verzí projektu. Ty jsou označeny pořadovým číslem pro lepší orientaci v archivu celého projektu. Složka verze projektu již obsahuje všechny soubory, se kterými uživatel přichází do styku při generování projektu, kam se doplňují i nově vygenerované soubory pro danou verzi.

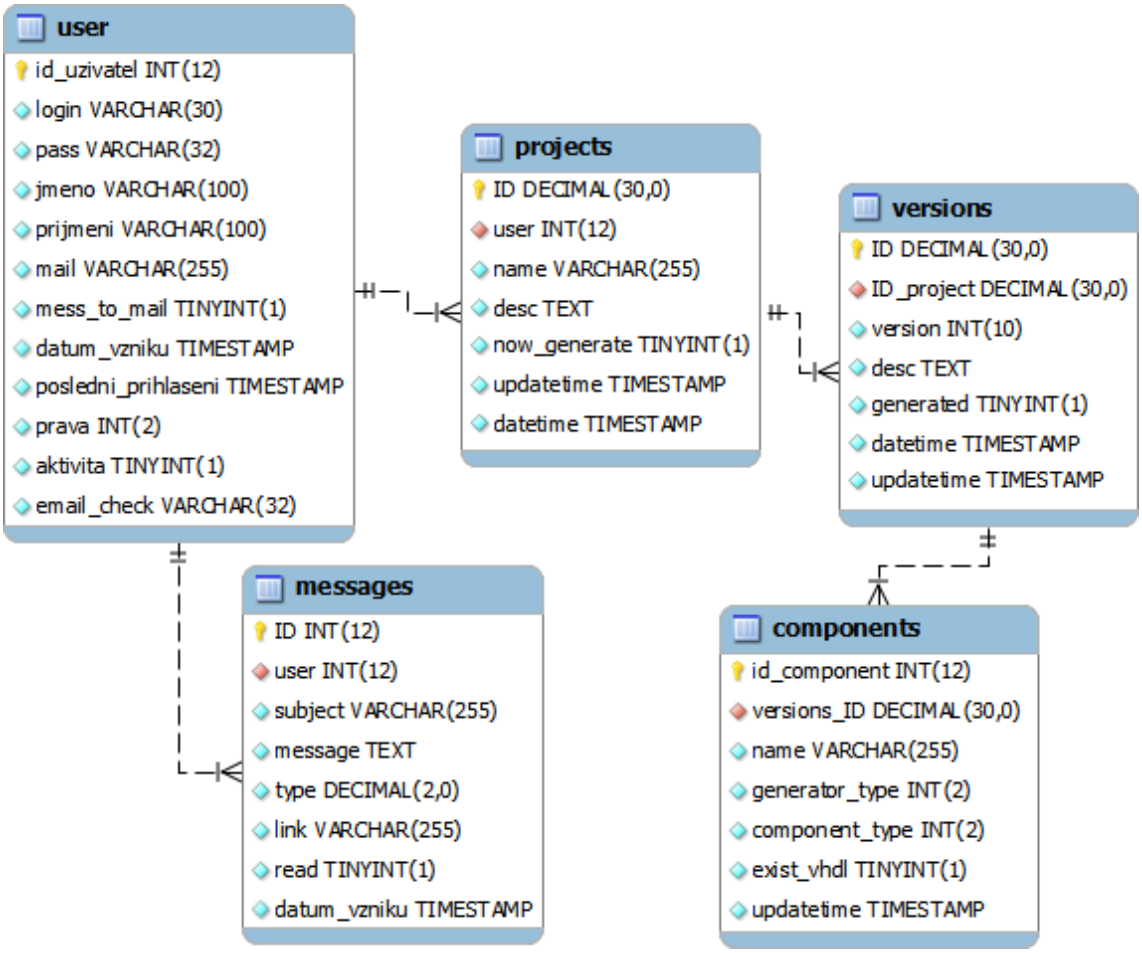
V názvech projektu se vyskytuje kromě začátečního označení `proj_` také identifikační číslo, tvořené časovou značkou vzniku projektu, na konci doplněné o identifikační číslo uživatele, který projekt vytvořil. Toto označení zajišťuje jednoznačné odlišení složek pro projekty, které mohou mít stejné jméno.

Veškeré identifikační údaje pro složky a soubory pak musejí být uloženy v databázi.

4.7 Entity–relationship diagram

Entity–relationship (zkráceně ER) diagram je nástroj k abstraktnímu a konceptuálnímu znázornění dat, používaný v softwarovém inženýrství. Tento diagram zobrazuje data, uložená v databázi a obsahuje typy entit a vztahy mezi nimi. Každá entita ER diagramu se vztahuje k určitému objektu reálného světa, což může být v naší aplikaci například uživatel, zpráva nebo projekt či verze určitého projektu.

Hlavní entitou je entita *user*, která obsahuje veškeré informace o uživateli a jeho nastaveních, jakým je třeba volba odesílání zpráv na e-mail. Na tuto entitu jsou přímo vázány dvě další entity, kterými jsou *messages* a *projects*. Entitu *messages* tvoří texty zpráv a jejich parametry, kterými je například příznak přečtení zprávy nebo odkaz na pokračování v projektu, o kterém zpráva informuje. Entita *projects* uchovává informace o projektu a váže na sebe entitu *versions*, týkající se informací o existujících verzích každého projektu. K této entitě je nejčastěji přistupováno, neboť identifikační číslo každé položky slouží jako adresa pro provedení úprav na dané verzi projektu. Poslední entitou, která se zabývá vygenerovanými komponentami, je entita *components*, ve které se uchovávají změny provedené ve čtvrtém kroku aplikace (změna typu generovaného kontrolního obvodu apod.). Diagram zobrazující vztahy mezi těmito entitami je na obrázku 4.5.

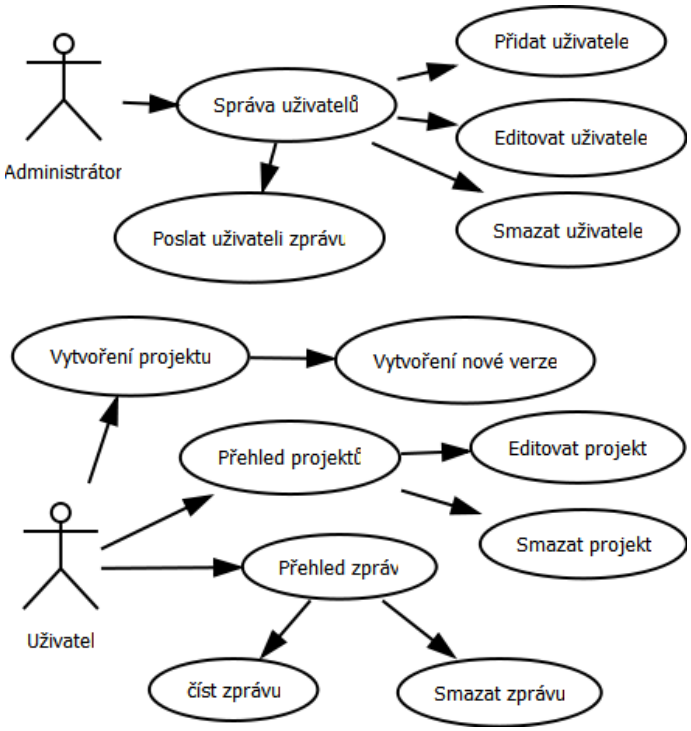


Obrázek 4.5: ER diagram dat uložených v databázi.

4.8 Diagram případů užití

Jedním z UML (Unified Modeling Language) nástrojů pro zachycení vnějšího pohledu na modelovaný systém je diagram případů užití (Use Case Diagram). Má za úkol pomoci odhalit hranice systému a slouží jako podklad pro odhady rozsahu srozumitelnou formou jak pro vývojáře, tak i pro uživatele.

Na obrázku 4.6 jsou vidět dvě odlišné role. Role administrátor je oddělena od role uživatel a má možnost provádět pouze změny v uživatelských údajích, či případně odeslat systémovou zprávu pro uživatele. Uživatelská role má oproti tomu přístup ke generování a editaci projektu spolu s možností prohlížení zpráv.



Obrázek 4.6: Diagram případů užití.

Kapitola 5

Implementace

Implementace rozebírá z hlavní části použité technologie a odůvodňuje, proč a kde byly uplatněny. Také se zabývá vysvětlením potřebných parametrů pro spuštění aplikace. V poslední kapitole je rozebrána instalace a kroky, které je potřeba provést.

5.1 Použité technologie

Základem funkčnosti webové aplikace je webový server, který zajišťuje zpracování HTTP (Hypertext Transfer Protocol) požadavků. Webových serverů použitelných pro tuto aplikaci je několik, například Sun Java System Web Server nebo Internet Information Services pro operační systém Windows. Nejznámějším webovým serverem, který byl použit i pro účely této práce je multiplatformní Apache HTTP Server. Ten byl vybrán pro snadnou konfiguraci a snadnou rozšiřitelnost ve stabilní verzi 2.

Ze zadání vyplývá nutnost použití programovacího jazyku PHP, jež je spolu s Apachem nejpoužívanější nástroj pro tvorbu webových aplikací. Pro uchování dat a uživatelských nastavení je potřebná plnohodnotná databáze, která zajistí možnost bezproblémového rozšíření aplikace oproti použití například XML souborů, které by tuto funkci bezproblémově zastaly. Databázový systém MySQL byl určen taktéž zadáním, nicméně patří mezi nejpoužívanější systémy plnící všechny potřebné funkce.

5.1.1 PHP

PHP (PHP: Hypertext Preprocessor, původně Personal Home Page) je skriptovací jazyk určený především pro tvorbu dynamického webu. Nejčastější začlenění patří přímo do struktur jazyků HTML a XHTML. PHP lze také využít pro tvorbu desktopových či konzolových aplikací. PHP je zcela nezávislý na platformě, skripty se nemusejí upravovat, aby fungovaly na více operačních systémech a jeho velkou výhodou je podpora mnoha rozličných knihoven pro různé účely — např. zpracování textu, grafiky, práci se soubory, přístup k většině databázových systémů (mj. MySQL) apod.

Díky jednoduchosti použití, ponechání vývojáři částečné svobody v syntaxi a kombinaci vlastností více programovacích jazyků se PHP rychle stalo velmi oblíbeným jazykem [4].

Protože PHP vyšlo již v několika verzích, bylo nutné se pro jednu z verzí rozhodnout. U každé použité funkce byla v manuálu zjištěna vyžadovaná verze, a následně vybrána pro aplikaci verze PHP 5 ze dvou důvodů. Prvním důvodem je úplná podpora objektově orientovaného přístupu, druhým důvodem je použití některých funkcí, které nižší verze nepodporují.

V této aplikaci bylo využito možnosti objektově orientovaného programování v PHP z důvodu použití malých podpůrných tříd, které jsem již dříve naprogramoval v jiných projektech. Všechny použité funkce jsou zahrnuty do patřičných tříd pro možnost znovupoužití a uloženy jsou ve složce `classes`. Nebyla však naprogramována žádná obecná třída, která by se dala použít v jiných případech bez jakýchkoli úprav.

Použité knihovny

V rámci PHP byla použita pouze knihovna *dibi*. Tato knihovna zajišťuje spojení mezi databází a PHP. Zajišťuje rychlý výběr dat z databáze a nabízí několik funkcí pro snadnější následnou manipulaci.

Knihovna je volně ke stažení na stránce http://dibiphp.com/cs/ a kopie licenční smlouvy, která je podmínkou použití této knihovny, je součástí distribuce aplikace.

5.1.2 Python

Python je velmi rychlý dynamický interpretovaný jazyk. Patří mezi skriptovací jazyky, které jsou šířeny jako otevřený software, tzv. open source. Je to program, který běží na mnoha platformách, a tak není problém s jeho přenositelností. Základní balík obsahuje také velké množství modulů, které lze bezproblémově zapojit do výsledného programu [5].

Tento jazyk byl vybrán pro pomocný program díky své rychlosti, bezproblémové manipulaci se soubory a možností lehce spouštět jiné programy.

Program v tomto jazyce pro výslednou aplikaci slouží jako spouštěč programů na pozadí. Princip tohoto programu byl popsán v kapitole 4.4.

5.1.3 HTML

HTML (HyperText Markup Language) je značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na internetu. Jazyk vychází z dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Další vývoj HTML byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka.

Pro ukončení vývoje HTML byla připravena verze 4.01. Další vývoj psaní dokumentů měla dle W3C (World Wide Web Consortium) patřit jazyku XHTML (následovník HTML, využívající univerzální jazyk XML). Pro nedokonalost vývoje okolo XHTML se část tvůrců webových prohlížečů, jako například Mozilla Foundation, Opera Software či Apple, rozhodlo založit iniciativu WHATWG (Web Hypertext Application Technology Working Group), jejímž cílem bylo vytvořit novou verzi HTML, která se posléze začala označovat jako HTML 5.[7]

V tomto jazyce je publikována pro webové prohlížeče celá aplikace. Každá stránka byla validována a opravena podle směrnic organizace W3C pro XHTML 1.0.

5.1.4 CSS

Kaskádové styly (Cascading Style Sheets) byly vyvinuty jako jazyk pro popis způsobu zobrazení jazyků HTML, XHTML nebo XML. Jazyk byl navržen standardizační organizací W3C, která zatím vydala již tři úrovně specifikace. Hlavním smyslem bylo umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. Původně to měl umožnit

už jazyk HTML, ale v důsledku nedostatečných standardů a konkurenčního boje výrobců prohlížečů se vyvinul jinak.[7]

V aplikaci jsou použity kaskádové styly pouze v externím souboru. Tato možnost nabízí změnu celkového vzhledu jednoduchou výměnou souboru bez nutnosti zasahovat do struktury v HTML. Taktéž bylo v HTML použito formátování stránky pomocí blokových prvků DIV, namísto formátování pomocí tabulek. Toto řešení nabízí flexibilnější úpravu stránky.

5.1.5 JavaScript

JavaScript je interpretovaný programovací jazyk, tvořící základ vývoje webových aplikací. Pomáhá dodat stránkám interaktivitu nebo v něm vytvořit celou aplikaci. JavaScript je na standardech založený jazyk s formální specifikací a ačkoli každý z dnešních webových prohlížečů interpretuje tuto specifikaci trochu jinak (z tohoto důvodu je těžší dosáhnout stejného výsledku pro všechny uživatele), většina webových prohlížečů používá základní funkce stejně. Při vývoji aplikací v tomto jazyce je potřeba klást důraz spíše než na vlastní použití Javascriptu na použití založeném na standardech.

JavaScript je nedílnou součástí využívané technologie AJAX (Asynchronous JavaScript and XML), která umožňuje načítání pouze požadovaného obsahu ze serveru, bez nutnosti načtení celé stránky.[11]

Ve čtvrtém kroku aplikace se uživateli nabízí několik komponent, kde je nutné nabídnout možnost volby pro každou z nich. Tento krok lze implementovat bez použití JavaScriptu a pro každou komponentu vypsat samostatnou stránku, což však obnáší nutnost zobrazit více stránek, z nichž se každá musí samostatně načíst. Druhou možností je u každé komponenty zobrazit rozbalovací okno, kde si uživatel vybrané parametry přímo zvolí bez přecházení na jinou stranu. Toto je možné jednoduše implementovat s použitím jazyka JavaScript a knihovny jQuery.

Knihovna jQuery

Lehká, malá javascriptová knihovna jQuery klade důraz na interakci mezi JavaScriptem a HTML. Je svobodným volně šiřitelným softwarem, který nabízí široké spektrum funkcí, z nichž aplikace využívá jen práci s efekty a animacemi.

5.1.6 MySQL

Aplikace vyžadovala použití databázového systému, který bude volně šiřitelný, multiplatformní, lehce ovladatelný ze strany PHP a zároveň rychlý ve zpracování požadovaných dotazů. Nabízela se možnost využití buďto MySQL nebo PostgreSQL. Nakonec bylo vybráno MySQL díky předchozím zkušenostem s jeho obsluhou.

MySQL je databázový systém, který nyní vlastní společnost Sun Microsystems a který je dostupný jak pod bezplatnou licencí GPL (GNU General Public License), tak pod komerční placenou licencí. Komunikace s databází probíhá pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o upravenou verzi tohoto jazyka s různými rozšířeními. Tento produkt lze instalovat jak na Linux, tak i na MS Windows, ale i na další systémy. MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušení.[4]

5.2 Požadované parametry

Pro správný běh aplikace je nutné mít správně nakonfigurovaný webový server pod operačním systémem Windows. Aplikace je omezena pouze na tento operační systém z důvodu spouštění programů, které jsou přeloženy pouze pro tento operační systém. Na tomto systému je v první řadě potřeba instalovat webový server Apache verze 2.2. Základní konfiguraci je potřeba rozšířit o funkčnost modulu `mod_rewrite`, který zajišťuje plnou funkci souboru `.htaccess` umístěného v kořenové složce projektu, který se stará o další změny v konfiguraci Apache. Dalším programem na instalaci musí být PHP ve verzi 5.2 a vyšší, kde není třeba po základní instalaci cokoli měnit. Databázový systém MySQL není omezen na verzi distribuce a taktéž stačí povést základní instalaci. Pro podporu generování kódů je potřeba nainstalovat Python 3 s podporou knihoven `os`, `sys`, `time` a `urllib2`.

Pro prohlížení aplikace je možné použít jakýkoli dostupný webový prohlížeč, držící se standardů zobrazování HTML podle W3C. Doporučeným prohlížečem je však FireFox či Chrome.

5.3 Instalace

Po splnění minimálních požadavků pro běh aplikace z předchozí kapitoly je nutné přesunout aplikaci do kořenové složky, kterou Apache zpřístupňuje na portu 80 pro webové prohlížeče. Aplikace musí být v kořenové složce z důvodu předávání parametrů oddělených znakem „/“, ve formě takzvaných přátelských URL (Uniform Resource Locator) odkazů, což zvyšuje přehlednost výsledné URL adresy.

Jako další bod instalace je změna parametrů uvnitř samotných souborů aplikace. Parametry, které je nutné změnit pro přístup do databáze, jsou umístěny v PHP třídě *Application* v souboru *Application.php* ve složce `classes/application/`. Zde je potřeba změnit název databáze, přístupové jméno a heslo ve funkci *_initDbConnection()*. Při použití knihovny `dibi` je možné tyto údaje jednoduše změnit v přehledně indexovaném poli, což ilustruje následující kód:

```
$options = array(  // localhost
            'driver'   => 'mysql',
            'host'     => 'localhost',
            'database' => 'jmeno_databaze',
            'username' => 'uzivatelske_jmeno',
            'password' => 'heslo',
            'charset'  => 'utf8',
);
```

Po upravení těchto parametrů je možné aplikaci spustit a potřebné tabulky i s vzorovými daty se do databáze vloží samy bez potřeby dalšího zásahu správce. Aplikaci lze spustit zadáním adresy souboru *index.php* do webového prohlížeče. Tato adresa může vypadat v případě použití na lokální stanici následovně: `http://localhost/index.php`.

Pro vstup do portálu pod uživatelským účtem je po instalaci zaveden účet s přihlašovacím jménem „test“ a heslem taktéž „test“. K administraci uživatelských účtů je potřeba se přihlásit s právy administrátora, k čemuž slouží účet se jménem „admin“, a stejně tak heslem „admin“.

Volitelným parametrem pro změnu nastavení je počet minut pro automatické odhlášení uživatele, které je možné změnit v souboru *index.php* v kořenové složce projektu. Počet

minut je jediným parametrem při volání třídy *Admin* a ve zdrojovém kódu se tato proměnná jmenuje `$autologout`, standardně nastavena na hodnotu 30.

Kapitola 6

Testování

Tato kapitola má za úkol čtenáře obeznámit s testováním, které probíhalo souběžně s tvorbou aplikace. V každém kroku tvorby aplikace byly odzkoušeny možné vstupy uživatele a registrovány odezvy aplikace, které byly následně v případě potřeby opraveny, či odladěny pro přijetí daného vstupu. Testováním aplikace v průběhu tvorby se předejde nutnosti změny velké části projektu v případě výskytu chyby v počátečních fázích vývoje. Tento způsob testování je však časově náročnější a v případě velkých projektů je nutné jednotlivé části projektu dostatečně oddělit, aby opravy jedné části neměli žádný vliv na jiné části projektu.

Funkčnost aplikace byla postupně rozšiřována s každou další dokončenou verzí. Problémy implementace byly rozděleny na malé celky, které řeší samostatné objektové třídy. Tyto třídy se skládají z jednotlivých funkcí, u kterých byla testována funkčnost vždy po dokončení. Vzájemné interakce těchto funkcí a obsluha tříd byla testována vždy po dokončení celé verze. První verze obsahovala například jen portál s možností přihlašování uživatelů a rozšířenou editací uživatelů. Na této verzi se stavěla všechna rozšíření až do výsledné podoby aplikace.

Testování probíhalo na osobním počítači s operačním systémem Windows 7, na kterém byly nainstalovány všechny potřebné programy. Aplikace neklade žádné zvýšené požadavky na systém, a pokud je v počítači nainstalován webový prohlížeč s grafickým prostředím, bude aplikace funkční.

Vzhled aplikace byl testován ve všech nejčastěji používaných prohlížečích (Windows Internet Explorer 8, Mozilla Firefox 3.6, Google Chrome a Opera 10) a jejich nejaktuálnější verzi vydání. V prohlížečích Internet Explorer a Opera je problém se správným zobrazením některých (validně napsaných) grafických prvků, což ovšem nemá dopad na funkčnost aplikace. Z tohoto hlediska je kompletní funkčnost zaručena na prohlížečích FireFox a Chrome, které jsou schopné zobrazit všechny validní prvky správně i za použití knihovny jQuery, která ovlivňuje některé jejich parametry.

Funkčnost aplikace byla testována na vzorovém archivu projektu, který je uložen na přiloženém CD ve složce `testovaci_data`, kde jsou zároveň uloženy i vzorové výstupy téhož projektu. V případě tohoto projektu je aplikace schopna vygenerovat všechny požadované hlídací architektury bez jakýchkoli chyb. Pro případ editace kódů je třeba dbát na správnost syntaxe, neboť generátor formálního popisu architektury není zatím odladěn na chybné vstupy a jeho běh se pro špatně zadaný vstup zacyklí, a tím je pozastaven i běh webového portálu.

Pro nasazení webového portálu na veřejně dostupný server je nutné tyto aplikace odladit i pro špatně zadané vstupní soubory. Zatím se počítá s využitím aplikace zkušenými uživateli, kteří jsou poučeni o funkčnosti programů. Hlavní výhodou i nadále zůstává možnost

hromadného generování hlídacích obvodů pro korektně zadané architektury.

Jako ukázkové kódy výstupu aplikace byly do složky s testovacími daty uloženy také archivy obsahující výsledky co nejodlišněji zadaných parametrů ve čtvrtém kroku aplikace (složka `testovaci_data/VYSTUP`). Prvním testovacím výstupem (složka `test1`) byla možnost, kdy uživatel zvolí u všech obvodů možnost generování TMR architektury. Možnost generování TMR architektury byla v druhém testovacím výstupu změněna na generování duplexní architektury (složka `test2`) a následně ve třetím výstupu (složka `test3`) byla zadána volba vygenerování hlídacího obvodu jen pro prvky, pro které existuje formální definice. Poslední, čtvrtý, testovací výstup (složka `test4`) obsahuje archivy, které jsou vygenerovány na základě výchozího nastavení bez změn provedených uživatelem, kde jsou předvolený možnosti podle rozpoznaných možností pro každou komponentu zvlášť. Pro každý ukázkový výstup je zavedena složka, obsahující archiv projektu, verze a jednoho vybraného prvku.

Jelikož po dobu testování byl portál spuštěn na lokálním serveru bez poštovních služeb, nebylo možné odesílat registrační e-maily na jakékoli adresy. Funkčnost tohoto bodu se však otestovala na zasílání zpráv do lokální stanice, kde vše proběhlo bez chyb a e-mail obsahoval oproti klasickému e-mailu pro veřejnou síť pouze upravenou adresu příjemce.

Testována byla taktéž i instalace portálu, kde se osvědčilo automatické vložení údajů potřebných pro chod aplikace. Aplikace je vázána na administrační účet, který pokud chybí, nelze vytvořit jinak, než přes webové rozhraní databáze MySQL. Díky automatickému importu důležitých dat do databáze v případě chybějících potřebných entit se zabrání chybovým hlášením při vybírání neexistujících údajů z databáze. Tento krok byl zaveden pro zjednodušení instalace správcem systému. Soubor pro import do databáze se nachází ve složce `installdb`, která je součástí zdrojových kódů.

Výsledný náhled aplikace je v příloze A, která slouží jako průvodce obsluhou portálu.

Kapitola 7

Možná rozšíření

Předposlední kapitola má za úkol přiblížit všechny možnosti dalšího rozšíření aplikace, které byly nalezeny při navrhování systému.

Při tvorbě čtvrtého kroku aplikace (volby parametrů jednotlivých komponent) vznikl problém, jak zobrazit celou architekturu v jediném kroku, kde si uživatel může zvolit komponenty k úpravě a zároveň vidí, se kterými dalšími komponentami je prvek spojen. V rozsahu této práce byl implementován textový výpis komponent pomocí seznamu, v němž je poskytnuta možnost zobrazení odkazu na komponenty, se kterými je prvek spojen. Pokud by se tento krok rozšířil o grafické prostředí, zobrazující architekturu se všemi komponentami a jejich propojeními, mohl by uživatel definovat parametry jednotlivých spojení a signálů a zároveň mít přehled o upravované architektuře. Za tímto krokem by mohla stát i další možnost rozšíření, a to například automatické generování formálních definic prvků, které je možné popsat pouze na základě znalosti propojení prvků. V předpokládaném grafickém prostředí by mohly být zobrazovány formuláře pro vyplnění údajů o komponentě dynamicky nad každou zvolenou komponentou, čímž by bylo rozvržení stránky úspornější.

Dalším možným rozšířením portálu může být rozšíření funkčnosti administrační sekce. Ta slouží zatím pouze k administraci uživatelů a zasílání systémových zpráv. Vhodným doplňkem by však mohla být možnost zobrazení statistik generování programů, které by mohly sloužit správci systému ke zjištění vytíženosti serveru a programů v průběhu dne.

S předchozími možnostmi rozšíření portálu bylo počítáno i při implementaci a všechny kroky jsou tvořeny pomocí samostatných objektových tříd, které je možné zaměnit za třídy, které budou obsahovat zmíněná rozšíření.

Kapitola 8

Závěr

Výsledný portál pokrývá všechny požadavky specifikace, které rozšiřují požadavky uvedené v zadání projektu. Systém byl doplněn o implementovaná rozšíření již ve fázi návrhu, čímž bylo s těmito úpravami počítáno již při první fázi implementace, a rozšíření tedy nejsou nadstavbou systému, ale přímo jeho součástí. Při implementaci nenastal žádný problém se specifikovaným zadáním a portál je připraven k reálnému použití v lokální síti. Testování odhalilo chyby v používaných generátorech kódů, nad kterými je aplikace vystavěna. Tyto chyby způsobují neukončené čekání na odpověď od serveru a na veřejně přístupném serveru by byl tento stav zcela nepřípustný.

Ve výsledku bylo dosaženo funkčního webového portálu, který uživateli nabízí možnost vložení vygenerovaných hlídacích obvodů, TMR a duplexních architektur přímo do projektu, který vloží v archivu na vstupu. Aplikace dovede uživatele během pěti kroků k výslednému archivu, kterým může nahradit svůj předchozí projekt, nabídnuta je taktéž možnost ukládat projekt v několika, na sobě nezávislých verzích. Nadstandardní možností u takovýchto aplikací je možnost od rozpracovaného projektu v kterémkoli kroku odejít s tím, že projekt bude uložen se všemi již provedenými změnami. Tato funkce nabízí uživateli možnost pracovat na projektu bez nutnosti svoji práci dokončit, pokud bude potřebovat k dalšímu kroku informace z jiného zdroje.

Plná funkcionalita systému je zaručena ve všech testovaných prohlížečích, přičemž jen ve dvou z nich je porušen vzhled aplikace u jednoho z pěti kroků (díky špatnému zobrazení některých prvků ze strany prohlížečů). Důraz byl kladen na validitu výsledných HTML kódů stejně jako na efektivnost prováděných PHP skriptů. Vzhledová stránka portálu je zaměřena na elegantnost spolu s moderními prvky vzhledu, přičemž je dbán důraz i na přehlednost a jednoduchost aplikace.

Z hlediska správce systému je portál lehce udržovatelný díky použití objektově orientovaného programování v jazyce PHP. Jednoduchou výměnou jediné třídy se změní funkcionalita bez výrazného vlivu na ostatní třídy. Primitivnější funkce jsou uloženy ve společné třídě, která nabízí možnost opětovného použití funkcí v jiných projektech.

Aplikace nabízí rozsáhlé možnosti rozšíření, čímž se zvýší možnosti použití. Další vývoj této aplikace tudíž může být tématem pro navazující diplomovou práci, která nabídne zpracování nejdůležitějšího rozšíření, kterým je grafické zobrazení architektury vloženého projektu. Také portál nabízí možnosti přidružení dalších nástrojů, které mohou vzejít z výzkumu, ze kterého dosavadní práce vychází.

Jazyková mutace portálu může být vhodným doplňkem v případě potřeby sdílet aplikaci i s univerzitami v jiných zemích. Pro tuto možnost by však bylo zapotřebí větších úprav v celém projektu.

Vývoj od prvotní analýzy požadavků po konečnou verzi projektu trval čtyři měsíce s tím, že čtvrtinu času zabralo testování aplikace s následnými opravami.

Literatura

[1] Catsoulis, J.: Fault Tolerance and Triple Modular Redundancy (TMR). [Online; navštíveno 03. 05. 2010].
URL http://www.embedded.com.au/pages/TMR.html

[2] Chu, P. P.: *RTL hardware design using VHDL: coding for efficiency, portability, and scalability*. John Wiley and Sons, 2006, ISBN 978-0-471-72092-8, 669 s.

[3] Elnozahy, E. M.; Melhem, R.; Mossé, D.: Energy-Efficient Duplex and TMR Real-Time Systems. In *RTSS '02: Proceedings of the 23rd IEEE Real-Time Systems Symposium*, Washington, DC, USA: IEEE Computer Society, 2002, ISBN 0-7695-1851-6, str. 256.

[4] Gilmore, J. W.: *Velká kniha PHP 5 MySQL: kompendium znalostí pro začátečníky i profesionály*, ročník 1. Zoner Press, 2005, ISBN 80-86815-20-X, 711 s.

[5] Harms, D. D.: *Začínáme programovat v jazyce Python*, ročník 2. Computer Press, 2008, ISBN 978-80-7372-378-1, 456 s.

[6] Hlavička, J.; Racek, S.; Golan, P.; aj.: Číslicové systémy odolné proti poruchám. 1992.

[7] Schafer, S. M.: *HTML, XHTML a CSS: bible [pro tvorbu WWW stránek]*, ročník 1. Grada, 2009, ISBN 978-80-247-2850-6, 647 s.

[8] Straka, M.: Generátor hlídacích obvodů pro komunikační protokoly Xilinx FPGA. In *Počítačové architektury a diagnostika 2007*, University of West Bohemia in Pilsen, 2007, ISBN 978-80-7043-605-9, s. 129–136.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=8441

[9] Straka, M.: Aplikace hlídacích obvodů v architekturách odolných proti poruchám. In *Počítačové architektury a diagnostika 2008*, Liberec University of Technology, 2008, ISBN 978-80-7372-378-1, s. 97–102.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=8698

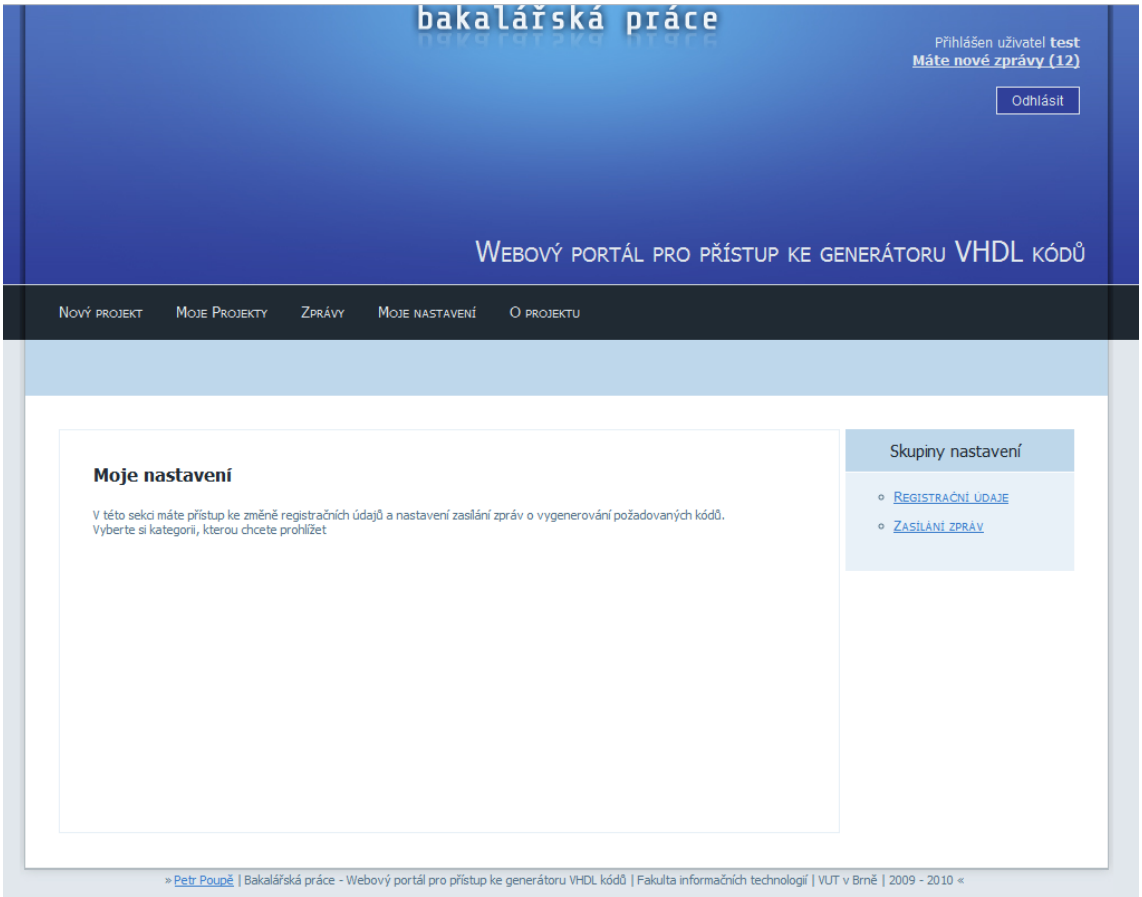
[10] Straka, M.; Tobola, J.; Kotasek, Z.: Checker Design for On-line Testing of Xilinx FPGA Communication Protocols. *Defect and Fault-Tolerance in VLSI Systems, IEEE International Symposium on*, ročník 0, 2007: s. 152–160, doi:http://doi.ieeecomputersociety.org/10.1109/DFT.2007.21.

[11] Suehring, S.: *JavaScript: krok za krokem*. Computer Press, 2008, ISBN 978-80-251-2241-9, 335 s.

Příloha A

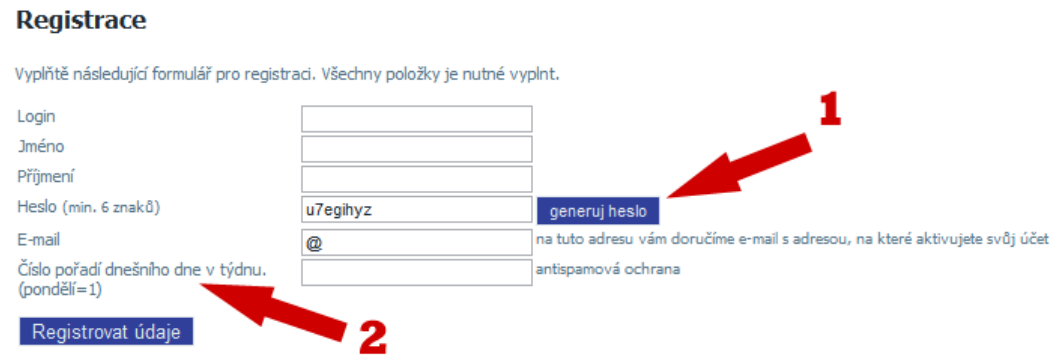
Uživatelův průvodce portálem

Prvním obrázkem (A.1) je reprezentován celkový vzhled aplikace při zobrazení uživatelských nastavení. V tomto kroku je zobrazeno i pravé postranní menu, které je použito jen na vybraných stránkách.



Obrázek A.1: Vzhled aplikace.

Registrace nových uživatelů probíhá pomocí formuláře na obrázku A.2. V tomto kroku je zavedeno pomocné generování bezpečného hesla, pomocí tlačítka pod číslem 1. Číslo 2 označuje antispamovou ochranu, která má za úkol znemožnit automatickou registraci do portálu pomocí stroje.

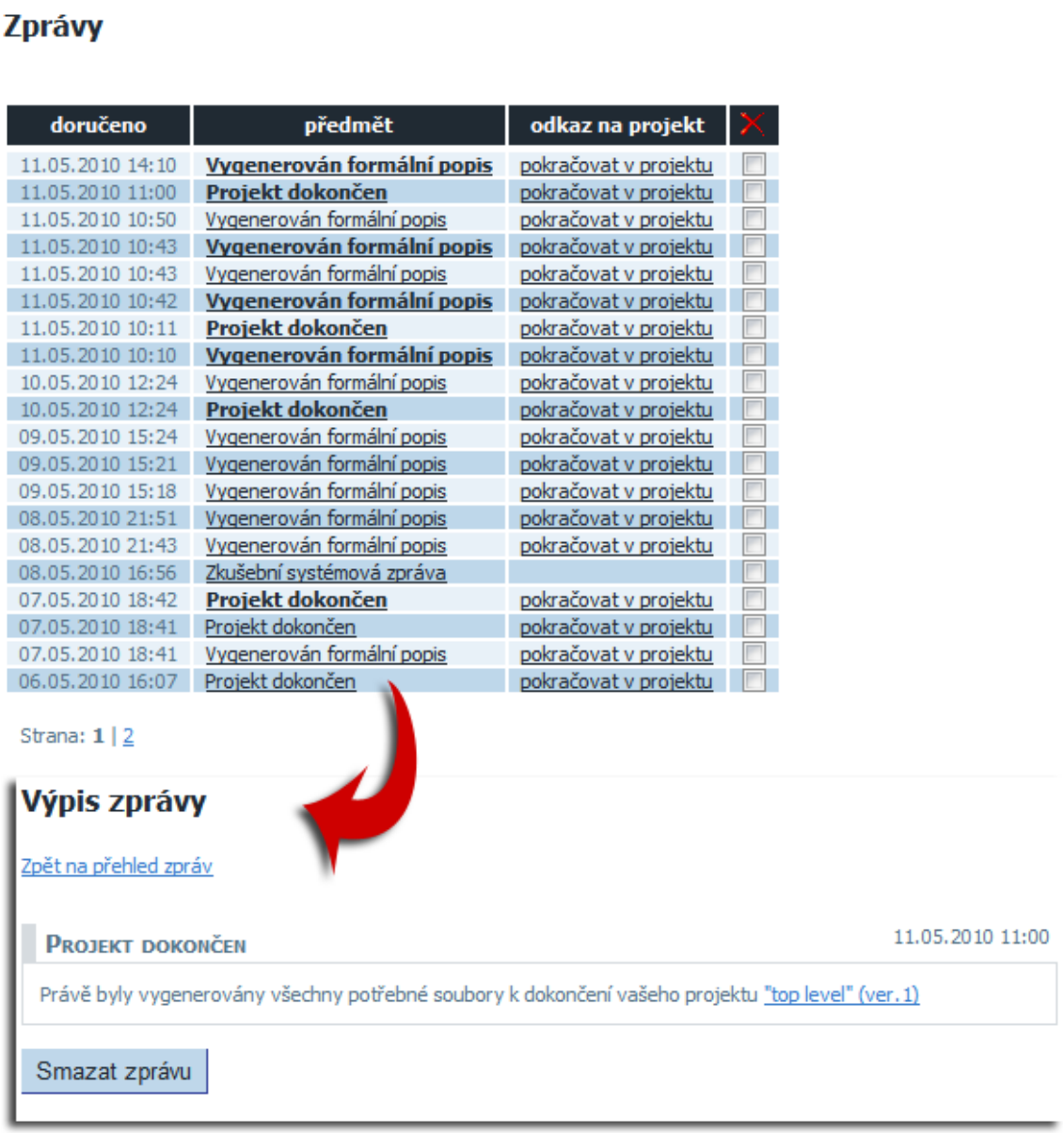


Obrázek A.2: Registrační formulář.

Obrázek A.3 uvádí pod číslem 1 přihlašovací dialog. Po přihlášení se na jeho místě zobrazí informace o počtu nepřečtených zpráv (ilustruje část označená čísly 2 a 3), spolu se jménem přihlášeného uživatele.

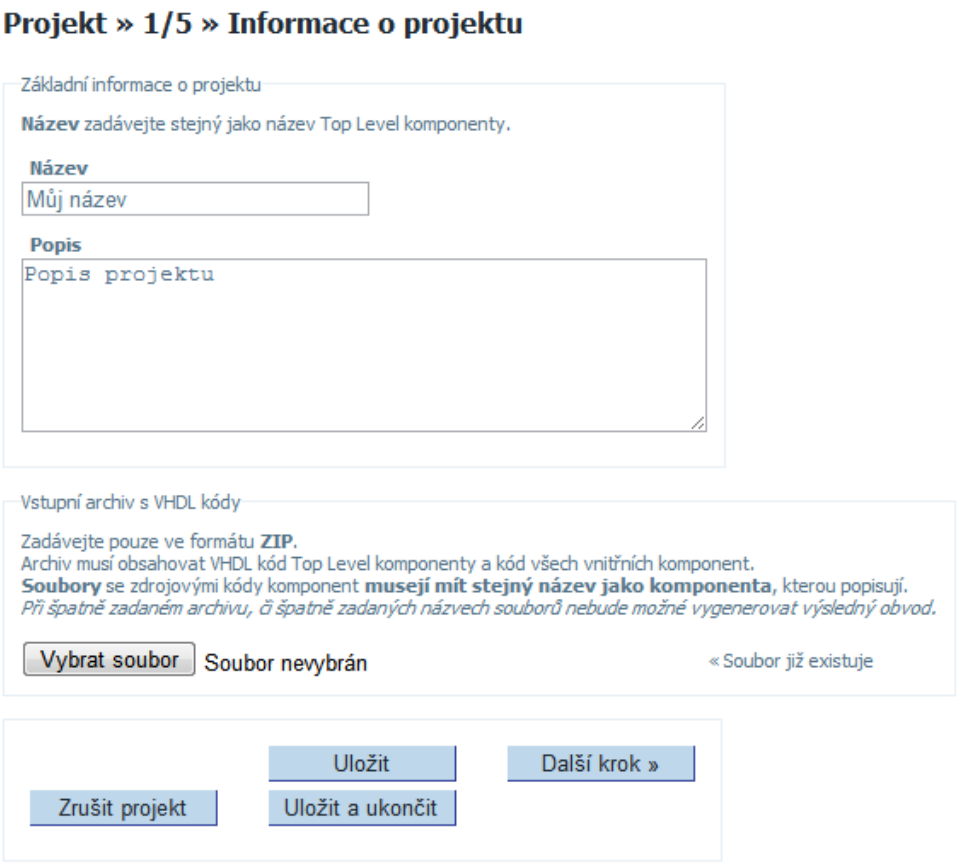
Obrázek A.3: Informace o počtu nepřečtených zpráv.

K přečtení zpráv je možné přistoupit buďto odkazem na hlavním menu, nebo pomocí odkazu informujícího o počtu nepřečtených zpráv. Seznam zpráv s ukázkou vygenerované zprávy po dokončení projektu je uveden na obrázku A.4



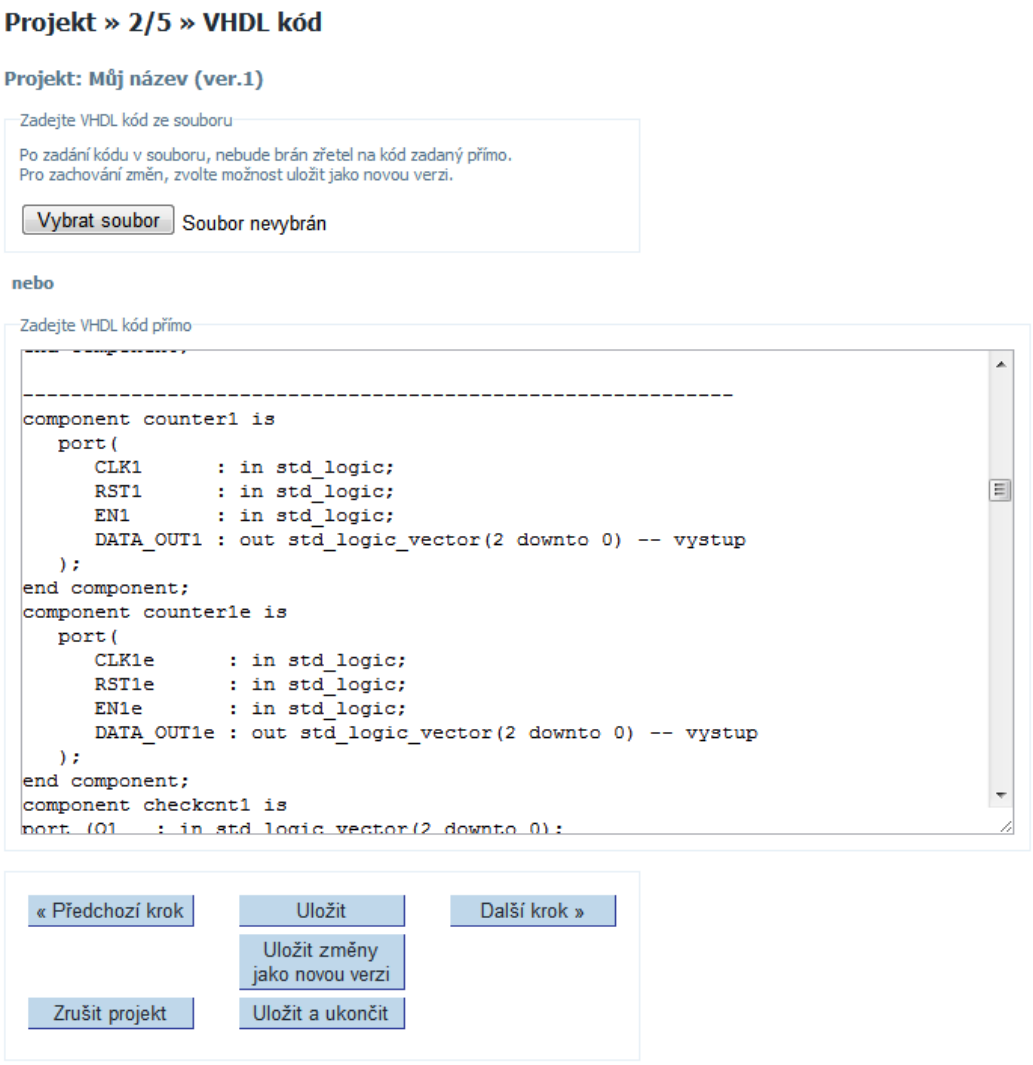
Obrázek A.4: Výpis zpráv s otevřenou zprávou.

Pro vygenerování nového projektu je zapotřebí projít pěti kroky, které jsou podrobně vysvětleny v technické zprávě projektu. Jako ilustrace těchto kroků slouží obrázky A.5 až A.9.



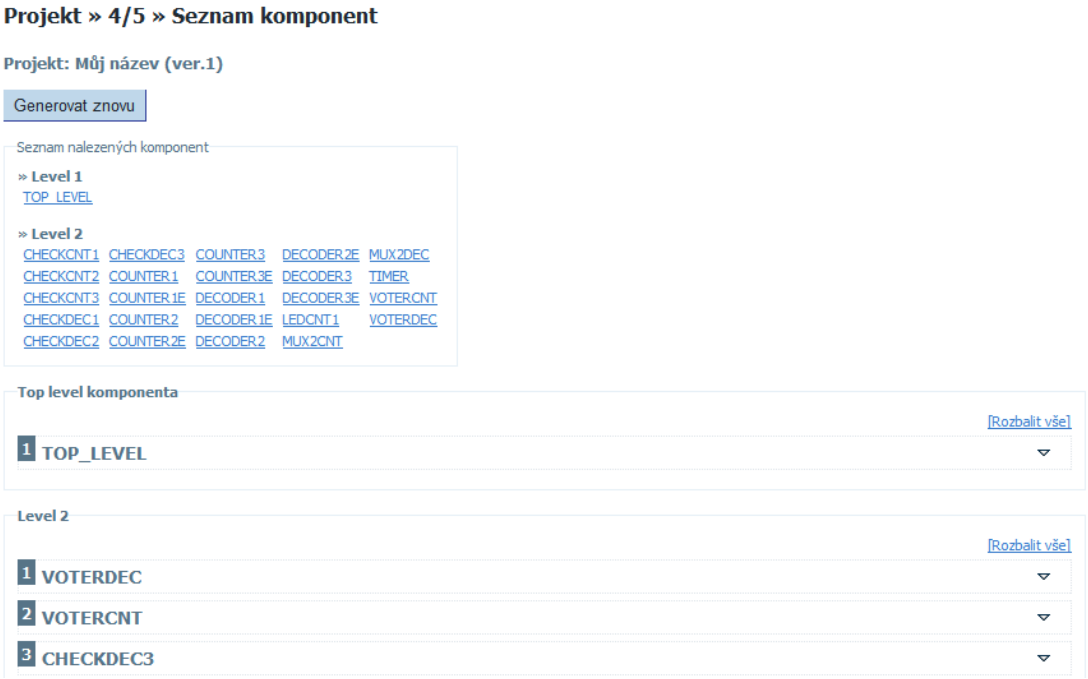
Obrázek A.5: První krok aplikace.

Seznam již vytvořených projektů (obrázek A.10) naleznete pod jednou z položek v menu, nazvanou „Moje projekty“. Číslo 1 ukazuje na odkaz s obrázkem koše. Tímto tlačítkem je možné po potvrzení volby celý projekt smazat. Číslo 2 na tom samém obrázku odkazuje na zobrazení dostupných verzí daného projektu.

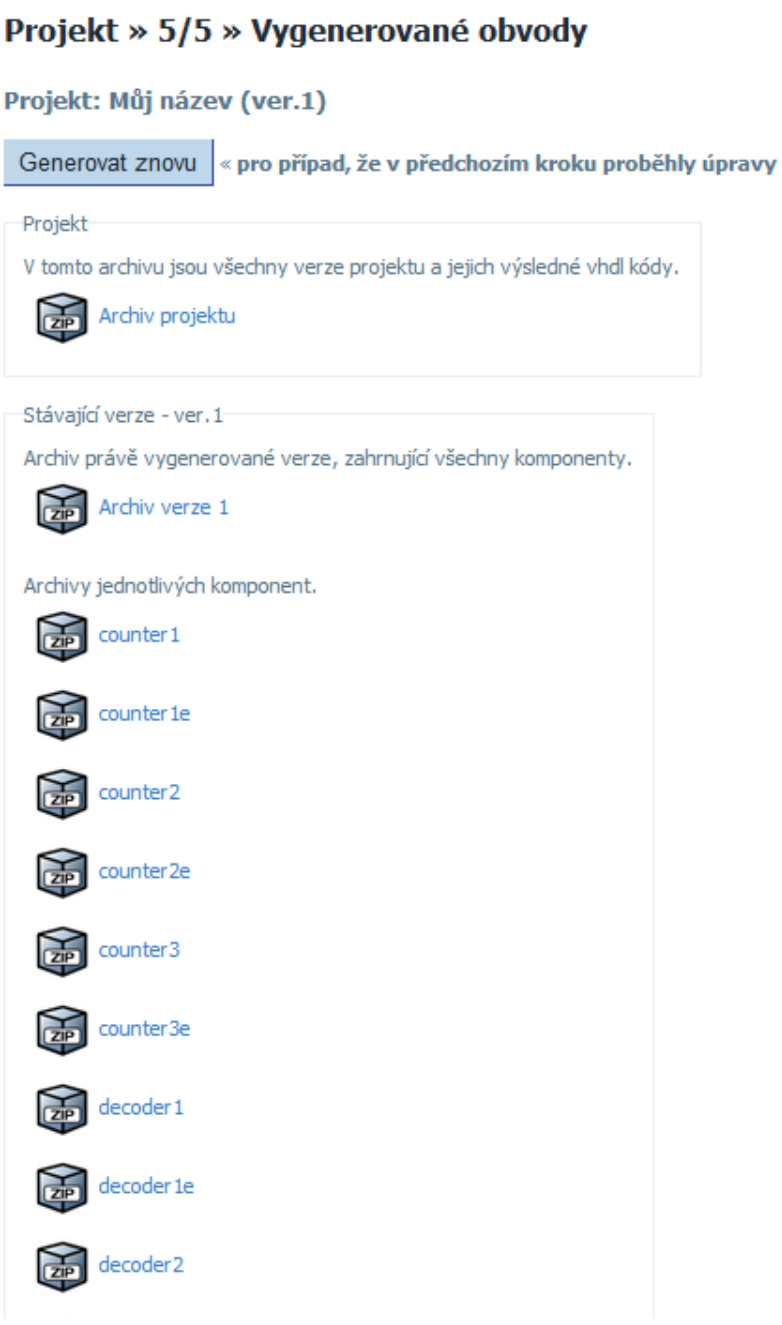


Obrázek A.6: Druhý krok aplikace.

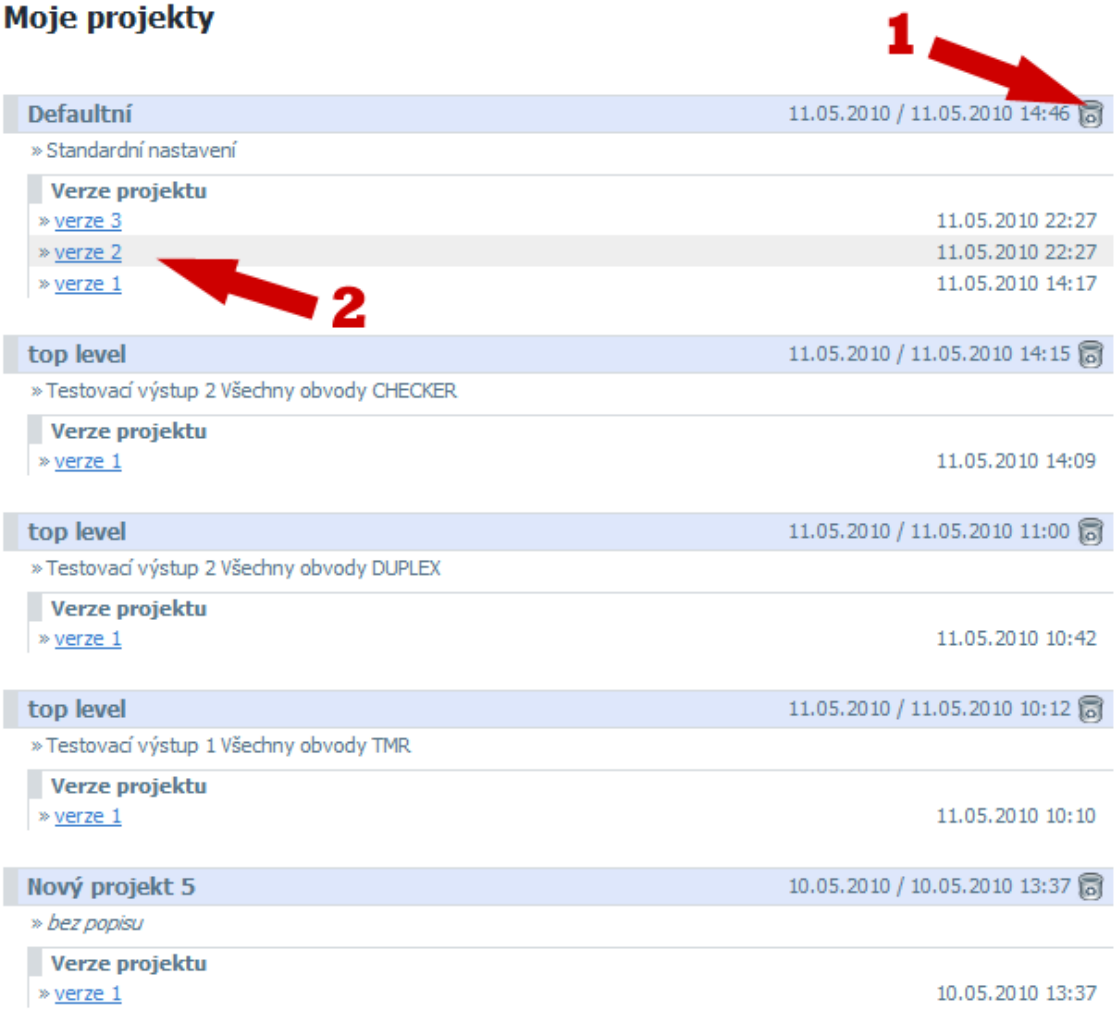
Obrázek A.7: Třetí krok aplikace.



Obrázek A.8: Čtvrtý krok aplikace.



Obrázek A.9: Poslední krok aplikace.



Obrázek A.10: Výpis již existujících projektů.

Příloha B

Struktura testovací architektury

Aplikace byla testována na obvodu pro spolehlivé řizení sedmisegmentového číselného displeje na desce ML506 s FPGA virtex5. Všechny soubory obsahují kódv jazyce VHDL.

Seznam VHDL souborů v archivu ZIP:

- counter1.vhd
- counter1e.vhd
- counter2.vhd
- counter2e.vhd
- counter3.vhd
- counter3e.vhd
- decoder1.vhd
- decoder1e.vhd
- decoder2.vhd
- decoder2e.vhd
- decoder3.vhd
- decoder3e.vhd
- checkcnt1.vhd
- checkcnt2.vhd
- checkcnt3.vhd
- checkdec1.vhd
- checkdec2.vhd
- checkdec3.vhd
- ledcnt1.vhd

- mux2cnt.vhd
- mux2dec.vhd
- timer.vhd
- top_level.vhd
- votercnt.vhd
- voterdec.vhd

Příloha C

Formální definice chování čítače

```
//entity count7 is
//  port(
//       CLK : in std_logic;
//       RST : in std_logic;
//       START: in std_logic;
//
//       COUNT: out std_logic_vector(2 downto 0)
//      );
//end count7;
@
p8=RST==0101 and START==1;
p9=RST==1 and START==0 and COUNT==000;
p0=COUNT==000 and RST==0 and START==0;
p1=COUNT==001 and RST==0 and START==0;
p2=COUNT==010 and RST==0 and START==0;
p3=COUNT==011 and RST==0 and START==0;
p4=COUNT==100 and RST==0 and START==0;
p5=COUNT==101 and RST==0 and START==0;
p6=COUNT==110 and RST==0 and START==0;
p7=COUNT==111 and RST==0 and START==0;
#
(S0,p0):S1;
(S1,p1):S2; (S1,p9):S0;
(S2,p2):S3; (S2,p9):S0;
(S3,p3):S4; (S3,p9):S0;
(S4,p4):S5; (S4,p9):S0;
(S5,p5):S6; (S5,p9):S0;
(S6,p6):S7; (S6,p9):S0;
(S7,p7):S0; (S7,p9):S0;
$
```

Příloha D

Obsah CD

Obsah CD je následující:

- Zdrojové kódy aplikace.
- Složka `testovaci_data` se vstupními testovacími daty a ukázkami výstupů aplikace.
- Technická zpráva ve formátu PDF.
- Technická zpráva ve formátu LaTeX.
- Soubor README.